Gameka: Uma ferramenta de desenvolvimento de jogos para não programadores

Igor Augusto de Faria Costa Alessandra Silva de Souza Carla Denise Castanho
Universidade de Brasília

Resumo

A crescente complexidade e sofisticação dos jogos eletrônicos têm impulsionado a elaboração de ferramentas de apoio ao desenvolvimento cada vez mais poderosas. Entretanto, aquelas mais comumente utilizadas, como engines e frameworks, exigem do desenvolvedor o domínio de conhecimentos de programação. A concepção de ferramentas que viabilizam o desenvolvimento de jogos para usuários não programadores constitui um desafio ainda em construção. Nesse contexto, este artigo apresenta a ferramenta *Gameka*, um software de código aberto e multiplataforma para o desenvolvimento de jogos 2D de gêneros diversos, que dispensa conhecimentos de programação. A ferramenta possui um ambiente de desenvolvimento completamente visual e disponibiliza ao desenvolvedor diversas funcionalidades prontas para a criação de jogos de maneira simples e intuitiva. Testes de usabilidade com um grupo de alunos do ensino médio da rede pública indicam que o software *Gameka* atingiu satisfatoriamente seus propósitos.

Keywords:: desenvolvimento de jogos, ferramenta de apoio, não programadores, editor de jogos, código aberto

Author's Contact:

{igor.augusto.ti, alessandrabuba}@gmail.com
carladenisecastanho@gmail.com

1 Introdução

O popularidade dos jogos eletrônicos tem despertado, principalmente nos mais jovens, o desejo e interesse na criação do próprio jogo. Ferramentas de apoio ao desenvolvimento, como engines e frameworks, ainda se apresentam como uma barreira para aqueles que não detém conhecimentos de programação [Rabin 2005].

Com o objetivo de viabilizar o desenvolvimento de jogos pelo público não programador, diversas ferramentas de apoio foram criadas. Por meio destes *softwares* é possível criar um jogo mesmo que o usuário possua pouco ou nenhum conhecimento de programação, simplesmente arrastando objetos para um cenário e determinando seus comportamentos através de menus. Dentre as ferramentas atualmente existentes, as mais relevantes no contexto deste estudo são: *MultimediaFusion* [Mul 2010] , *Game Maker* [Gam 2010b], *RPG Maker VX* [RPG 2010], *Game-Editor* [Gam 2010a], *GameSalad* [Gam 2010c], *JumpCraft* [Pla 2011], *Kodu* [Kod 2010], *MyGame-Builder* [MyG 2010], *BYOND* [BYO 2010], *Alice* [Ali 2010], *Venatio Creo* [Wolbach et al. 2009] e <e-Adventure> [Torrente et al. 2010].

Em geral, uma ferramenta de apoio ao desenvolvimento de jogos para não programadores possui, pelo menos, três funcionalidades: um editor de objetos, um editor de cenários e um editor de eventos. Dessa forma, o usuário pode criar objetos, selecionando sua representação gráfica e determinando seu comportamento. Estes objetos podem ser posicionados em um cenário de forma a montar o ambiente (mundo) do jogo. Por fim, podem ser criados eventos, que são um conjunto de cláusulas nas quais determina-se uma condição e uma ação a ser executada caso a condição seja verdadeira.

Dentre as ferramentas mencionadas acima muitas possuem foco no desenvolvimento de um tipo específico de jogo como, por exemplo, *RPG Maker VX* (RPG) ou *Plataform Studio* (Plataforma). Outras, como a BYOND, dispõem de um conjunto restrito de funcionalidades prontas disponíveis ao desenvolvedor. Além disso, muitas são comerciais, a exemplo da *Multimedia Fusion 2*, *GameSalad*, e *RPG Maker VX*, ou não são multiplataforma, como *Game Maker* e *Kodu*, o que restringe sua difusão, utilização e portabilidade.

Neste contexto, este trabalho apresenta a ferramenta *Gameka*, um *software* de código aberto e multiplataforma para o desenvolvimento de jogos 2D de gêneros diversos, que dispensa a codificação de programas. A ferramenta contém uma interface simples e diversas funcionalidades que facilitam a criação de jogos, como tipos de movimentação de objetos prontos para serem utilizados, editores de fundos e animações, um criador simplicado de objetos e um editor de eventos visual. Com o objetivo de validar os propósitos da ferramenta desenvolvida foram conduzidos testes de usabilidade com um grupo de alunos do ensino médio da rede pública de educação.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 é apresentada uma análise comparativa das principais ferramentas de apoio ao desenvolvimento de jogos para não programadores. A Seção 3 apresenta a ferramenta *Gameka*, suas principais características e funcionalidades. Detalhes de implementação e testes de validação estão, respectivamente, nas Seções 4 e 5. Por fim, na Seção 6 são apresentados considerações finais e trabalhos futuros.

2 Trabalhos correlatos

A Tabela 1 apresenta uma análise das doze ferramentas de desenvolvimento de jogos para não programadores mencionadas na seção anterior. Para cada uma, foram avaliados os seguintes quesitos: (i) licença (paga/gratuita), (ii) necessidade de conhecimento de programação (sim/não), (iii) a ferramenta permite programação (sim/não), (iv) número de funcionalidades prontas (baixo/médio/elevado), (v) grau de flexibilidade (baixo/médio/alto), (vi) gênero de jogo a que destina a ferramenta, e (vii) plataforma (Windows/Linux/Mac/Web).

Todos os softwares analisados possuem um editor de cenários, através do qual é possível construir um cenário e nele posicionar objetos para que possam interagir entre si no jogo. Observa-se também que grande parte das ferramentas analisadas são comerciais. Estas são as que oferecem um número maior de funcionalidades prontas, o que facilita significativamente o desenvolvimento de jogos. Dentre os softwares gratuitos analisados, aquele que mais se aproxima das alternativas comerciais em termos de funcionalidades é o *Game-Editor*.

Dentre as alternativas pesquisadas, apenas a *BYOND* necessita conhecimentos de programação, sendo que as demais possuem um editor visual de eventos, através do qual é possível definir o comportamento dos objetos através de menus. Em geral, escolhe-se uma condição e uma ação, a partir de uma lista limitada de possibilidades, que será ativada se a condição for verdadeira. Contudo, algumas variações de complexidade podem ser observadas. A ferramenta *Kodu Game Lab*, por exemplo, trata a movimentação de um personagem da seguinte forma: "se o analógico esquerdo do controle do controle for utilizado, então mover", enquanto outras ferramentas tratam essa movimentação através incremento da posição do personagem no eixo x.

Mesmo que o editor de eventos seja suficiente para desenvolver um jogo, algumas ferramentas apresentam a possibilidade de programação através de *scripting*, tais como *Game-Editor*, *RPG Maker VX*, *Game Maker* e *Platform Studio*. É importante ressaltar que esta alternativa exige conhecimentos básicos de programação.

Uma característica fundamental neste tipo de software de apoio é o suporte avançado à construção do jogo por meio da disponibilização de funcionalidades prontas que tratem de aspectos de movimentação, colisão, arte, inteligência artificial, fundos, animações, além de particularidades para tipos específicos de jogos. Considera-se como

	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

Tabela 1: *Comparativo das ferramentas para apoio ao desenvolvimento de jogos.*

tendo um número "elevado" de funcionalidades prontas as ferramentas que contemplam a maioria dos aspectos acima. Quando no máximo dois aspectos são tratados pela ferramenta assume-se que o número de funcionalidades é "baixo", enquanto que "médio" foi atribuido àquelas que contemplam um número intermediário de aspectos. O *Multimedia Fusion 2*, por exemplo, já possui pré-programada a detecção de colisão e disponibiliza vários tipos de movimentação prontos, tais como, oito direções, veículo de jogos de corrida 2D e tipo plataforma. Entretanto, estes objetos de movimentação são relativamente limitados e dispõem de poucas funcionalidades para editá-los adequadamente, podendo ocasionar erros de execução no jogo produzido.

A ferramenta *Game-Editor* possui uma quantidade intermediária de funcionalidades prontas. Apesar de fornecer opções para gerenciamento de colisão e animações, ela não permite que se atribua um tipo pré-determinado de movimentação a um objeto além da movimentação por caminhos, que envolve a determinação de pontos a serem seguidos em série pelo objeto. Dessa forma, torna-se necessário que o usuário determine, através do editor de eventos da ferramenta, as ações e condições necessárias para que o objeto se movimente. Para criar um objeto que se move em oito direções de acordo com os botões pressionados, por exemplo, faz-se necessária a criação de todos os eventos de deslocamento nos eixos x e y para cada uma das direções pressionadas no teclado.

Além de facilitar o trabalho do desenvolvedor, uma ferramenta de apoio ao desenvolvimento de jogos deve viabilizar ao máximo a implementação das ideias de concepção do jogo, evitando que estas necessitem adequar-se às limitações da ferramenta. Desta forma, o software deve ser flexível, ou seja, permitir a expansão das características das entidades do jogo através do editor de eventos. Conforme mencionado acima, o *Multimedia Fusion 2* disponibiliza um objeto que se movimenta em oito direções pronto para ser utilizado. No entanto, através da criação de eventos, a ferramenta permite que o usuário crie seu próprio objeto caso não esteja satisfeito com o objeto padrão. Nesse sentido, as ferramentas *Game Maker*, *Game-Editor*, *GameSalad*, *BYOND* e *Alice* são consideradas flexíveis. O software *Kudo* oferece uma flexibilidade média, permitindo que sejam criadas novas funcionalidades mas sem muitas variações, enquanto que *MyGameBuilder* é relativamente pouco flexível, uma vez que os desenvolvedores devem se restringir às funcionalidades disponibilizadas.

3 Ferramenta *Gameka*

A ferramenta *Gameka* é um *software* de apoio ao desenvolvimento de jogos 2D voltado ao público não programador, cujas principais características são: *open-source*, multiplataforma e direcionada ao desenvolvimento de gêneros diversos. A ferramenta possui um ambiente de desenvolvimento completamente visual e disponibiliza ao desenvolvedor diversas funcionalidades prontas para a criação dos mais variados gêneros de jogos 2D, tais como aventura, *shooter*, nave, corrida e *puzzle*. Por meio de um assistente de criação de objetos, a *Gameka* automatiza o processo de definição de características das entidades do jogo, permitindo uma rápida configuração inicial de vários aspectos de jogabilidade.

3.1 Interface principal

A *Gameka* fornece uma interface (Figura 1) que tem como recurso principal um editor de mapas cujo visualizador encontra-se no centro da tela. Ele interage com um seletor de *tiles* e outro de mapas posicionados à sua esquerda, além de um seletor de objetos à sua direita. Este posicionamento pode ser alterado pelo usuário.

O visualizador de mapas permite a inserção, movimentação e remoção de *tiles* e objetos no/do cenário através da integração com os seletores da interface. O seletor de *tiles* possui uma grade de seleção de *tiles* e um seletor de camada atual. A partir dele, é possível selecionar, com o *mouse*, um *tile* individualmente ou em agrupamentos adjacentes e pintá-los na camada do mapa previamente escolhida. O seletor de mapas permite, a partir de uma lista, a seleção de mapa para edição, além da criação de novos mapas. O seletor de objetos disponibiliza uma lista de objetos para serem instanciados e posicionados no mapa, além de um botão para criação rápida de objetos. Este botão ativa uma janela (Figura 2) onde é possível criar um objeto preenchendo um pequeno formulário com as características como tipo, velocidade de movimentação, representação gráfica, estilo de movimentação e habilidades. A idéia é permitir a criação de um objeto de forma simplificada, sem necessidade de utilização de menus complexos.

3.2 Configurações Gerais

A janela de Configurações, onde são determinados os comportamentos gerais do jogo, está subdividida nas seguintes abas: (i) iniciais, (ii) objetos, (iii) *Tilesets*, (iv) animações, (v) fundos e (vi) eventos. Detalhes de cada configuração são descritos a seguir.

3.2.1 Configurações Iniciais

Na aba de configurações "Iniciais", detalhes comuns do jogo são determinados, tais como nome do jogo, resolução da tela, mapa inicial, imagens para telas do jogo, músicas das telas principais e detalhes de fim de jogo.

3.2.2 Configurações de Objetos

A aba "Objetos" apresenta configurações acerca das classes de objetos a serem utilizadas no jogo, permitindo inserção, configuração, e viabilizando suas posteriores instanciações nos mapas. A interface apresenta uma lista de objetos e um formulário dinamicamente construído para edição de aspectos específicos da classe selecionada na lista. Dentre os tipos de objetos possíveis estão: *Personagem*, que é um objeto controlado pelo jogador, *NPC* (*Non-Player Character*) *aliado*, que é um objeto capaz de emitir um diálogo na tela quando o personagem do jogador se aproxima dele; *NPC inimigo*, que é capaz de causar dano em um outro objeto, atirando

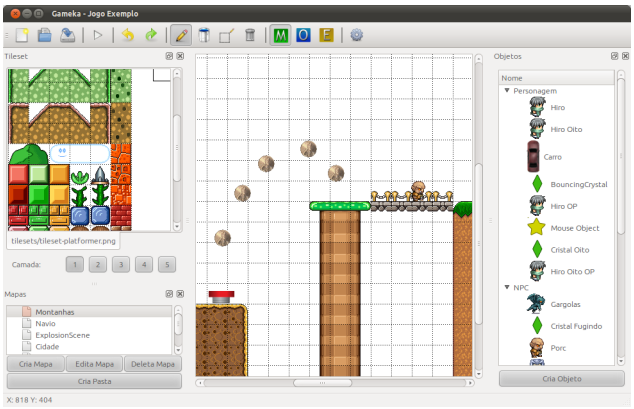

Figura 1: *Tela principal da Gameka.*

Figura 2: *Criador de objetos simplificado.*

Itens ou encostando em um objeto; e, por fim, *Item*, que pode ser coletado e/ou atirado, causando um determinado efeito ao encostar em um outro objeto.

É possível associar ao objeto do tipo *Personagem* diversas formas de movimentação, dentre os quais: movimento em oito direções, na qual o objeto se movimenta nas oito direções da tela de acordo o botão de seta do teclado pressionado; movimento de plataforma, em que o objeto se submete a uma gravidade que faz com que ele tenda a se movimentar para baixo até encontrar um obstáculo, podendo pular e se movimentar no eixo horizontal de acordo com a entrada do teclado; movimento de veículo, no qual o objeto gira para os lados ao pressionar os botões ← e → do teclado, acelera ao pressionar o botão ↑ e engata marcha ré ao apertar o botão ↓; movimento de acordo com o mouse, em que o objeto se movimenta seguindo o ponteiro do mouse sob determinadas condições; e estático, em que o objeto se mantém parado.

3.2.3 Configurações de *Tilesets*

A tela de configurações de *Tilesets* é responsável pelo gerenciamento de *Tilesets* a serem utilizados nos mapas. Podem ser disponibilizados diversos *Tilesets*, que fornecem uma variedade visual na construção dos cenários. Cada *Tileset* possui um nome, uma imagem associada, e configurações como tamanho do *tile* e o seu mapa de colisão, ou seja, a propriedade de cada *tile* de ser colidível ou não por objetos previamente definidos. Isto permite a detecção e resolução automática de colisão durante a execução jogo.

3.2.4 Configurações de Animações

Cada objeto poderá ter uma ou mais associações de animações, desde que estas sejam importadas e configuradas na tela de animações. É possível importar imagens contendo quadros em série de uma animação (*sprite sheets*) e configurá-las adequadamente de

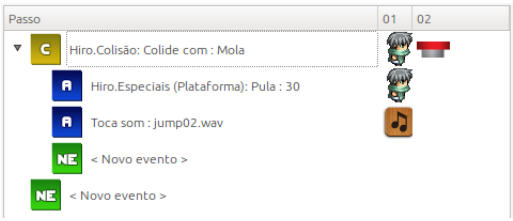

Figura 3: *Visão parcial do editor de eventos.*

modo a gerar uma animação no jogo. Dentre outros atributos, é necessário especificar o número de quadros na horizontal e na vertical, além do espaçamento em píxeis entre os quadros. É possível ter uma pré-visualização da animação.

3.2.5 Configurações de Fundos

Uma imagem de fundo é aquela que preenche as regiões de um mapa que são semi-transparentes ou não possuem *tiles*. Um fundo pode ter uma ou mais camadas, cada uma delas com uma imagem em uma determinada posição que pode se repetir horizontalmente e/ou verticalmente e possuir um determinado comportamento. Este comportamento, que tem relação com a posição do objeto principal da cena, permite a criação de efeitos visuais de horizonte como profundidade e movimentação do cenário.

3.2.6 Editor de Eventos

O objetivo do editor de eventos (Figura 3) é proporcionar flexibilidade à ferramenta, fornecendo um editor visual de algoritmos, cujo uso não requer a digitação de linhas de código. Na *Gameka* os eventos são orientados a procedimentos. Por padrão, são três procedimentos globais, dois deles ativados no início e no final do jogo, e o outro a cada quadro de jogo. Cada classe de objeto também se associa a três procedimentos, dois deles ativados na sua criação e destruição, e um deles ativado a cada quadro de jogo para cada uma das instâncias da classe de objeto que estiverem presentes no cenário. Também há a possibilidade de se criar novos procedimentos globais ou associados a objetos, os quais poderão ser chamados em outros procedimentos, em propriedades de objetos ou em diálogos.

Um procedimento é uma lista de eventos a ser executada. Cada evento pode ser uma condição, ação ou diálogo. Condições correspondem a testes dentro dos quais uma outra lista de eventos pode ser encadeada, a qual será acessada caso a condição testada em tempo de execução seja verdadeira. Já ações se relacionam a mudanças efetivadas no ambiente de jogo ou nos objetos. Por fim, diálogos são mensagens apresentadas na tela durante o jogo.

A inclusão de um novo evento em um procedimento é feita através de duplo clique no ícone “<Novo Evento>”. Um novo menu mostrará as opções “<Nova condição>”, “<Nova ação>” e “<Novo Diálogo>”. Cada uma dessas opções ativa uma janela na qual todas as possibilidades disponíveis pelo editor são mostradas, permitindo a construção dos eventos.

4 Implementação

A ferramenta *Gameka* está dividida em duas partes principais, o *Editor* propriamente dito, que também é responsável por gerar e gravar em disco um arquivo contendo todas as informações do projeto construído, e o *Runtime*, programa responsável por ler e interpretar este arquivo, e de acordo com as informações lidas, mostrar na tela e executar o jogo criado. A implementação da *Gameka* foi feita utilizando a linguagem *C++* e o paradigma de orientação a objetos. Para a implementação da interface gráfica da ferramenta, utilizou-se o *Qt* [Qt: 2010], um *framework* multiplataforma de interface gráfica para o usuário (GUI). As informações do jogo construído são armazenadas em um arquivo de extensão `.gmk`, o qual será lido posteriormente pelo *Runtime*. Para o armazenamento dos eventos em arquivo é utilizada uma estrutura de *bytecode*. Na implementação do *Runtime* foi utilizada a biblioteca SDL [SDL 2010].

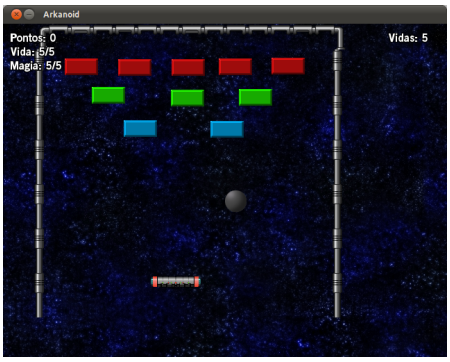

Figura 4: *Jogo similar ao Arkanoid.*

Figura 5: *Jogo Urban Runner.*

5 Validação

De acordo com *Sommerville* [Sommerville 2008], a validação de software destina-se a mostrar se um sistema está em conformidade com sua especificação de requisitos e atende às expectativas de seus usuários, estando assim aceito para o uso operacional. A ferramenta *Gameka* foi validada por meio da condução de uma oficina com um grupo de dez alunos do Ensino Médio da rede pública de educação que não possuiam qualquer conhecimento de programação. Foram realizados cinco encontros com duração de duas horas cada. Dividiu-se a atividade em três fases: (i) uma etapa inicial, na qual os alunos foram submetidos a um tutorial com o objetivo de familiarizá-los com a ferramenta; (ii) uma segunda etapa na qual os participantes, com o auxílio de um tutorial, desenvolveram um jogo semelhante ao *Arkanoid* [Fujita 1986]; e, por fim, (iii) um momento para os alunos criarem livremente um jogo utilizando a ferramenta. Foram aplicados quatro questionários ao longo do experimento, um no início, para conhecer o perfil do grupo, e outros três ao final de cada etapa, com o objetivo de avaliar a experiência de utilização da ferramenta. Para as duas primeiras fases, foram elaborados, como material didático, roteiros de utilização da ferramenta, que foram entregues impressos aos participantes. A utilização da ferramenta *Gameka* deu-se de maneira individual.

Com relação à facilidade de uso da ferramenta, os questionários respondidos indicaram que as atividades de associação de um objeto a uma animação e de criação de uma habilidade foram consideradas pelos participantes como as mais fáceis. A criação de animações também foi considerada fácil. Por outro lado, a operação de encontrar telas e botões apresentou-se como a de maior dificuldade. No contexto específico de desenvolvimento do jogo semelhante ao *Arkanoid* (Figura 4), os participantes relataram que as operações de criação dos blocos, da bola, da barra e do fundo, apesar de pouco detalhadas no roteiro, foram as mais fáceis da etapa. Como esperado, as atividades que envolveram o uso do editor de eventos foram consideradas as de maior dificuldade, uma vez que os participantes ainda não estavam habituados com a elaboração de regras envolvendo a lógica do jogo. Os sujeitos da pesquisa auto-avaliaram os jogos criados atribuindo notas de 0 a 10. A nota média foi 7,6.

O resultado da última etapa foi satisfatório tendo em vista que o grupo conseguiu desenvolver alguns jogos, na sua maioria do gênero plataforma ou aventura, dentre os quais detacaram-se *Urban Runner* (Figura 5) e *Luige*. A nota média de satisfação dos participantes com os jogos criados individualmente nesta fase foi de 8,1. O último questionário também mostrou que a nota média atribuída à ferramenta *Gameka* foi 9,6.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou a ferramenta *Gameka*, um *software* multiplataforma de código aberto para o desenvolvimento de jogos 2D de gêneros diversos, que dispensa a necessidade de conhecimentos de programação. Através de uma interface simples e intuitiva são fornecidas diversas funcionalidades prontas, além de um criador simplificado de objetos e um editor de eventos puramente visual. Testes de usabilidade com usuários sem conhecimento de programação indicaram que os propósitos da ferramenta foram atingidos. Trabalhos futuros incluem, dentre outros: desenvolvimento de novas funcionalidades; implementação de laços de repetição e funções no editor de eventos; e implementação do módulo *Runtime* utilizando tecnologias como *HTML5* e *JavaScript*, para permitir a execução dos jogos desenvolvidos em um navegador *web*.

Referências

2010. Alice: An Educational Software that teaches studentes computer programming in a 3D environment. `http://www.alice.org/`. Acessado em julho/2011.

2010. BYOND: Build Your Own Net Dream. `http://www.byond.com/`. Acessado em julho/2011.

FUJITA, A., 1986. Arkanoid. Arcade. `http://en.wikipedia.org/wiki/Arkanoid`.

2010. Game-Editor: The Cross Platform Game Creator. `http://game-editor.com`. Acessado em julho/2011.

2010. Game Maker 9. `http://www.yoyogames.com/gamemaker/`. Acessado em julho/2011.

2010. GameSalad: Game Creation for the Rest of Us. `http://gamesalad.com/`. Acessado em julho/2011.

2010. Kodu Game Lab. `http://research.microsoft.com/en-us/projects/kodu/`. Acessado em julho/2011.

2010. Multimedia Fusion 2: Ultimate game creation and powerful applications. `http://www.clickteam.com/eng/mmf2.php`. Acessado em julho/2011.

2010. MyGameBuilder. `http://mygamebuilder.com/`. Acessado em julho/2011.

2011. JumpCraft: The Easiest Way to Create a Game. `http://jumpcraft.com/`. Acessado em julho/2011.

2010. Qt: Cross-platform application and UI framework. `http://qt.nokia.com/`. Acessado em julho/2011.

RABIN, S. 2005. *Introduction to Game Development*. Charles River Media.

2010. RPG Maker VX: Role Playing Game Maker. `http://www.rpgmakerweb.com`. Acessado em julho/2011.

2010. SDL: Simple Directmedia Layer. `http://www.libsdl.org/`. Acessado em julho/2011.

SOMMERVILLE, I. 2008. *Engenharia de Software*. Pearson Education.

TORRENTE, J., BLANCO, A., MARCHIORI, E. J., MORENO-GER, P., AND FERNÁNDEZ, B. 2010. <e-Adventure>: Introducing Educational Games in the Learning Process. *IEEE EDUCON 2010 Conference* (Abril).

WOLBACH, E., STONE, L., AND KONTOSTATHIS, A. 2009. Venatio Creo: The Game Development Platform for Non-Programmers. *`http://www.venatiocreo.com/`* (Junho).