Relatos de caminho trilhado para o desenvolvimento de um jogo de adventure no Flash

Laiza Camurugy Arivan Bastos Lynn Alves Josemar Souza

Universidade do Estado da Bahia, Departamento de Educação, Brasil

Resumo

O presente artigo visa abordar o processo de desenvolvimento do jogo de *adventure* Búzios – Ecos da Liberdade, com foco na sua implementação. Neste trabalho registramos também as dificuldades e soluções encontradas para concluir e disponibilizar o jogo com conteúdos históricos para os cenários escolares, preservando a essência e característica básica dos jogos comerciais: ser divertido.

Palavras-chave: games, desenvolvimento, design

Contato dos autores:
laiza@dcc.ufba.br
lynnalves@yahoo.com.br
arivanbastos@gmail.com
josemarsbr@gmail.com

1. Introdução

Recentes estudos e projetos apontam os jogos eletrônicos na aprendizagem como tendo grande potencial para atingir a geração atual de "nativos digitais", ou seja, sujeitos que vivenciam diariamente a interação com as novas mídias como os videogames, e-mail, chat, telefones celulares e outras tecnologias interativas [Moita, 2009].

O último projeto do grupo de pesquisa Comunidades Virtuais, o jogo Búzios – Ecos da Liberdade, possibilita a mediação do aprendizado dos conceitos históricos, tendo como foco a Revolta de Búzios, também conhecida como Conjuração Baiana, ou Revolta dos Alfaiates. Este movimento abolicionista e de independência, ocorreu na Bahia em 1798, e se inspirou na Revolução Francesa (1792) e seus ideais: Fraternidade, Liberdade e Igualdade. O enredo, aliado ao conteúdo histórico da Revolta de Búzios, trata da escravidão e do contexto econômico, político e social. Alguns dos NPCs (Non-Player Character, ou personagem não jogável) são personagens históricos conhecidos, tais como Cipriano Barata, Lucas Dantas, Manuel Faustino e Luiz Gonzaga, importantes líderes da revolta.

O estilo adventure foi escolhido para ser adotado pelo jogo. Adventure Games ou Jogos de Aventura, são aqueles em que o jogador assume o papel de protagonista em uma história interativa, incentivando o raciocínio lógico por meio da solução de puzzles ou quebra-cabeças, e incentivam a exploração do cenário, o qual possui objetos específicos que precisam ser encontrados. Podem ser considerados como exemplos de adventure games os jogos Where in the World is Carmen Sandiego?, The Secret of Monkey Island, Full Throttle, dentre muitos outros.

As tecnologias utilizadas foram o Adobe Flash CS4 Professional, o Adobe Air 1.5 e o MDM Zinc 3.0, e a linguagem de programação para o desenvolvimento foi ActionScript 3.0.

2. Game Design

O processo de game design foi conduzido por uma equipe multireferencial, com membros de Pedagogia, História, Design e Artes, Programação e Música. O que faltou foi um especialista em game design, que tivesse experiência no desenvolvimento de jogos, e um amplo conhecimento das diversas áreas que compõe o grupo. Afinal, é o game designer que dita o direcionamento do jogo, coordena o que deve e o que pode ser feito, desde a parte de projeto, até a finalização do game. Talvez o próprio processo de criação tenha pulado passos importantes, como prototipagem do jogo para estudo de jogabilidade e possíveis caminhos a seguir. Construir protótipos do produto final permite testar atributos mesmo que este ainda não esteja pronto [Hom, 1998].

O modelo de game design adotado pelo grupo foi focado em organizar mais as *quests*, o número de cenários, e pensar todos os elementos desde o início. Apesar disso, a equipe ficou um pouco presa a um modelo em que a história define a mecânica e o gameplay, deixando o jogo limitado pelo enredo.

As fases de testes para o lançamento de versões foram de suma importância. Pudemos listar *bugs* e solucioná-los, acrescentando no nível de aprendizado do grupo. Dentre os bugs mais sérios, haviam os que provocavam o fechamento do jogo e os que impediam o avançamento das fases. Temos como exemplo os próprios erros de programação, ou a má formação da lógica nos arquivos XML.

Os jogos de adventure avaliados durante o estudo de similares, e que contribuíram no processo de Game Design do Búzios, foram Monkey Island, Full Throttle, Conspiração Dumont, The Dig, e Runaway.

Os mini-games que ocorrem durante o jogo foram peças importantes na sua dinâmica, na medida em que possibilitaram integrar diferentes gêneros em um único título: dentro do Adventure Búzios – Ecos da Liberdade temos um puzzle, o Torre de Hanói; um musical baseado no atual sucesso Guitar Hero, chamado Berimbau Hero; e um jogo casual, o Queda-de-braço. Estes mini-games introduzidos no enredo são necessários para completar as quests e proporcionam maior dinamicidade, motivando os jogadores a continuarem e vencerem os desafios propostos.

3. Concepção do roteiro

Para a construção do roteiro do jogo Búzios, foram necessários estudos da época em que ocorreu a Revolta, para que o jogo trouxesse um conjunto rico de informações. Foi feito um levantamento de todos os tipos de fontes, desde textos de autores consagrados da historiografia baiana, visitas a museus, arquivos públicos, bibliotecas, filmes, etc. Pesquisas por meio de filmes tinham como objetivo juntar subsídios necessários para a construção dos cenários e vestimenta dos personagens. É importante ressaltar que a equipe de roteiro foi composta de profissionais da área de Pedagogia e História.

As informações da história factual foram misturadas ou diluídas com outras informações ficcionais. O protagonista Francisco Vilar conduzido pelo jogador, torna-se um herói baiano que sai de Salvador para estudar direito em Coimbra – Portugal. No enredo do jogo, ele volta para Salvador para ver seu pai Vicente Vilar que está muito doente, em um momento de agitação social. O personagem logo se posiciona frente às mudanças que estão para acontecer.

Toda a equipe teve a preocupação e cuidado na forma de veicular as informações históricas, os diálogos e as posturas dos personagens, principalmente Francisco, por conta da importância histórica desta revolta para a Bahia, que valoriza, preserva e difunde a cultura afro-descendente.

4. Implementação

4.1 Arquitetura do Motor

Um motor de jogo possui uma arquitetura de software capaz de interligar uma série de componentes e propiciar uma experiência gráfica interativa em tempo real para os usuários [Bittencourt e Osório, 2006]. O motor do jogo Búzios foi todo desenvolvido no Flash, utilizando de seus recursos.

Enquanto os cenários não estavam prontos, foi feito um protótipo para testes das interações ‘pegar’, ‘observar’, e ‘falar’ e do localizador de rotas. A localização de rotas foi feita por meio de um mapeamento, no qual as áreas caminháveis são definidas por meio de um MovieClip. Nele, as áreas preenchidas são as áreas de colisão, e as não preenchidas são as áreas que se pode caminhar. A partir desse MovieClip, é criada uma matriz de colisão, utilizada como base para um algoritmo de localização de rotas A* pathfinding. Tal algoritmo utiliza uma função heurística para determinar a qualidade de cada um dos estados possíveis, por meio de um custo para a melhor rota até o destino passando pelo nó atual. Esse custo determina a qualidade do caminho. Quanto menor o custo, melhor é o caminho. [Perucia et. al., 2005]. Durante a inicialização das fases, o motor faz uma varredura do MovieClip de colisão, realizando testes de existência de preenchimento a partir da função hitTestPoint() oferecida pelo Action Script 3. Essa varredura gera uma matriz que representa o cenário. Por exemplo:

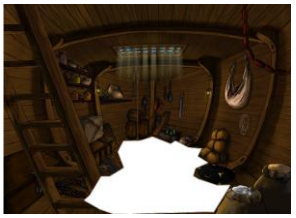

Figura 4: Cenário do navio com destaque para área caminhável

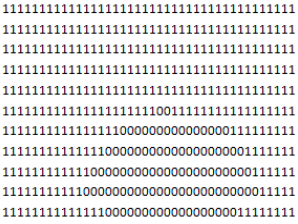

Figura 5: Matriz do cenário do navio

Quando o jogador clica em um ponto do cenário, o sistema calcula o elemento da matriz correspondente, fazendo uma relação entre a largura e altura do cenário e da matriz. Posteriormente, realiza a localização de rotas entre o ponto atual do personagem e o ponto destino, retornando um array de pontos pelos quais o personagem deve passar para chegar ao destino (todos pontos na matriz). Cada ponto presente no array é convertido então em coordenadas do cenário, e o personagem é então movido seqüencialmente entre esses pontos.

O roteiro do jogo foi implementado a partir de arquivos XML. Documentos neste formato contêm dados rodeados por tags (ou marcações). Neste caso, as tags defininem a estrutura e o significado dos dados. O motor desenvolvido possui uma classe chamada Interpretador, que faz o mapeamento do jogo através da leitura desses arquivos. Nas pastas de fase do jogo, há três tipos de arquivos XML:

- O arquivo da cena, o qual define suas características, como o arquivo “.swf” que a representa, suas dimensões, seus objetos não-interagíveis, ações iniciais, entre outras;
- Os arquivos dos objetos da cena, que definem seus nomes, seus MovieClips e ícones de inventário associados, as interações que o jogador terá com os objetos, etc.
- Os arquivos dos personagens da cena, com a mesma estrutura que os arquivos dos objetos, além de guardar todo diálogo que se pode ter com o personagem.

As interações com objetos e personagens podem ser de quatro tipos: pegar, falar, observar ou combinar. Para que uma interação e suas respectivas ações sejam executadas, é necessário que suas condições sejam satisfeitas. Por exemplo, ao pegarmos em uma porta, podem acontecer duas ações: ou podemos conseguir abri-la, ou seremos informados de que a porta está trancada. Se estivermos com a chave, o que seria uma condição, conseguiríamos abrir a porta,. Se não tivermos a chave, poderia haver um disparo de diálogo, onde o protagonista falaria "Está trancada.". A condição de estar com a chave possui valores de verdadeiro ou falso. Esses valores podem ser alterados ao longo do jogo, acarretando diferentes ações. É necessário guardá-los, pois ao sair de um cenário, e depois voltar a ele novamente, é necessário que ele esteja da mesma forma que o deixamos. O motor do jogo também se encarrega desse armazenamento de informações. Ao inicializarmos um cenário, essas informações também são inicializadas.

O protagonista explora os cenários em uma perspectiva 3D, já que, quanto menor for a sua posição Y no palco, menor ele ficará, e quanto maior sua posição Y maior ele ficará, dando uma noção de que o personagem se aproxima ou se afasta de uma câmera existente. Esse redimensionamento é realizado a partir de mudanças nas propriedades scaleX e scaleY do MovieClip do personagem. Além disso, ao caminhar pelo cenário, o personagem pode passar pela frente ou por trás de um mesmo objeto. Para isso, uma comparação é feita entre as coordenadas Y dos pontos de registro do personagem e dos objetos. Caso o personagem esteja em uma posição Y menor que a de um objeto (no Flash o Y cresce de cima para baixo), significa que ele está em um ponto mais distante em relação à perspectiva. Logo, ele deve ficar atrás do objeto. Caso contrário, ele deve ficar na frente por estar mais próximo. Durante a movimentação do personagem, as trocas de profundidade entre ele e os objetos são feitas a partir da função swapChildren() oferecida pelo Action Script 3.

4.2 Padrões de projeto

Os padrões de projeto (ou design patterns) utilizados foram o Singleton e o Observer.

O Singleton é utilizado para garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso para a mesma [Gamma et. al., 2005]. No nosso caso, seriam as classes Diretor e Interpretador, que possuem apenas uma instância de si próprias, durante a execução do jogo. A classe Diretor possui referências às demais classes que implementam a lógica do jogo, integrando suas funções.

Já o Observer define uma dependência um-para-muitos entre objetos, de modo que quando um objeto muda de estado, todos seus dependentes sejam notificados, e atualizados automaticamente. [Gamma et. al., 2005]. Ou seja, a partir do sistema de eventos do Flash, objetos podem se tornar "ouvintes" de ações ou mudanças de estado. Por exemplo, se ao clicarmos em um botão quisermos que aconteça algo, utilizaríamos neste botão o método "addEventListener", passando como parâmetro o tipo do evento, que seria um MouseEvent.CLICK, e a função que seria executada após o clique. Classes que utilizam o disparo de eventos estendem "EventDispatcher", tais quais as classes criadas no nosso motor: ObjetoCena, InterfaceDialogo, Inventario e Interpretador.

4.3 Desenho, pintura e animações

Para os personagens, a criação de arte conceitual foi feita no Adobe Photoshop, com base nas fichas criadas pela equipe de roteiro. Após essa etapa, foi iniciada a criação e animação quadro a quadro dos personagens em vetor no Flash. Finalizando, os personagem eram exportados para o formato PNG, e reorganizados no Flash, na fase de Level Design.

Quanto aos cenários, as artes conceituais foram feitas à mão. A depender do artista e do cenário, o traço finalizado era feito à mão ou diretamente no computador, através da tablet (ou mesa digitalizadora). Logo depois os cenários eram pintados digitalmente no Photoshop, e por fim as camadas eram mescladas e assim era exportado para o formato JPG. No Level Design, montava-se como background.

As animações foram feitas em camadas, as ilustrações à mão, pintadas no Photoshop, exportadas em camadas separadas, montadas e animadas no Flash. Os personagens e os cenários foram produzidos em 2D, mas foram feitos estudos em 3D para análise de perspectiva e animação.

4.4 O uso do Zinc

O Zinc é um aplicativo que auxilia no desenvolvimento para Adobe Flash. Dentre suas funcionalidades, temos a API {mdm}Script 3.0, com aproximadamente 500 novos métodos e classes específicos para aplicações Desktop, inclusive permite a leitura e gravação de arquivos, o que foi essencial para o save game. O Zinc permite alterarmos configurações da aplicação, como o tamanho da janela (tendo a opção de full screen), borda, ícone, ocultar ou não botões de maximizar, restaurar e fechar. Além disso, permite a criação de arquivos executáveis para Windows e Linux que independem da existência do plugin Flash Player no computador, facilitando a sua execução em ambas plataformas.

4.5 Principais dificuldades técnicas

- Desempenho com vetores (não pudemos manter as animações como vetores, pois o excesso de objetos no formato vetor na tela deixava o jogo extremamente lento).

- Limite na quantidade de quadros das animações (passamos a usar bitmaps para os objetos, mas uma grande quantidade de quadros bitmap nos objetos animados causava instabilidade no Flash durante o processo de level design).
- Leaks de memória, ou vazamento de memória, que ocorrem quando objetos não mais referenciados pelo programa em execução vão sendo acumulados na memória até que não haja mais espaço de armazenamento [Barros, 2006], causando um mau desempenho.

5. Avaliação

Foram feitas três pesquisas para validação do jogo por um público externo.

A primeira foi realizada em 2009 com professores da rede pública que não tinham expertise com os jogos eletrônicos, destacando-se apenas um que já tinha experiência. A segunda pesquisa ocorreu nos meses de maio e junho de 2010, com alunos do quinto ano do Ensino Fundamental de uma escola municipal. E a terceira pesquisa foi realizada nos meses de maio a julho de 2010, com 12 alunos do curso de extensão "Conteúdos interativos e Educação: construindo novas práticas pedagógicas", realizado no Departamento de Educação da Universidade do Estado da Bahia. O resultado dessa validação foi muito importante para o processo de desenvolvimento, pois subsidiados pelos feed-back obtido, realizamos alterações no jogo a fim de torná-lo mais próximo do universo dos professores, mas sem perder o caráter "*funny*" presente nos games comerciais.

6. Conclusão

Juntamente com o resultado obtido, tivemos alguns problemas.

O desempenho não foi o esperado. Quanto mais objetos, animações e quanto maiores eram os MovieClips, mais o Flash tornava-se lento e instável. O Flash não possui um bom mecanismo para gerenciamento de memória. Todos os quadros de animações vão para a memória, independente de estarem sendo utilizados na hora, ou não. Por exemplo, mesmo que um MovieClip esteja parado no seu primeiro quadro, todos os outros também vão para a memória. Esta não é uma situação ideal, e em um motor de jogo provavelmente não aconteceria, ou seja, os quadros só iriam para a memória somente quando fossem utilizados. Isto atrapalhou tanto na etapa de level design, como no desempenho do jogo em execução. Logo, para que não tivéssemos um jogo pesado, e com desempenho ruim, foi preciso simplificar e reduzir a quantidade de animações, bem como o tamanho de alguns cenários. Apesar deste não ser um caminho desejável, foi a única solução encontrada para executar o jogo nos espaços escolares, que apresentam equipamentos aquém das performances desejadas para jogar.

Além disso, percebemos a facilidade que o Flash tem em criar "leaks" de memória. Pela liberdade que o jogador tem de explorar os cenários em um jogo de Aventure, seu motor precisa carregar e descarregar cenas da memória com freqüência. Quando carregamos uma cena, é necessária a criação de '*event listeners'* para capturar ações, como por exemplo, os cliques do jogador sobre objetos. O principal problema do ActionScript 3 nesse aspecto, é que ele não oferece nenhum recurso para forçar a remoção completa de um filme (nesse caso, o cenário) da memória da aplicação em execução. Caso um filme seja removido e exista alguma referência que ainda aponte para algum elemento dele, o filme continuará na memória. Logo, para evitar os "leaks" de memória no Flash, precisamos assegurar que não existe mais nenhuma variável apontando para algum elemento do filme que deseja-se descarregar, e que todos os '*event listeners'* criados foram destruídos.

A compatibilidade do jogo com o Linux não foi completa. O Zinc não permite a execução da aplicação para Linux em tela cheia, e não bloqueia a sua maximização. Assim, foi necessário inserir uma máscara para que o jogador ao maximizar a tela não visse detalhes periféricos que não estivessem enquadrados pela câmera. Além disso, é necessário assegurar-se que o driver da placa de vídeo esteja instalado, caso contrário o jogo não funciona em sua normalidade. Por fim, o Zinc não permite a integração das fontes (tipografias) utilizadas ao executável do jogo. É necessário instalar as fontes usadas no Linux onde o jogo será executado. Desconsiderando-se as dificuldades mencionadas, a execução do jogo em ambiente Linux foi bastante facilitada pelo Zinc: o jogo rodou de forma estável e semelhante ao ambiente Windows, onde foi desenvolvido.

Apesar das dificuldades e restrições apontadas nesse artigo, percebemos que o grupo de pesquisa avançou no processo de desenvolvimento de games, agregando valor aos seus produtos e pesquisa, contribuindo para a qualificação profissional dos estudantes, profissionais e pesquisadores envolvidos no desafio de produzir jogos para os cenários pedagógicos, mas atentando para os aspectos que seduzem os jogadores nos games comerciais.

Agradecimentos

Agradecemos a Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB), pela oportunidade de irmos além da teorização e concretizar o sonho de desenvolver um jogo eletrônico, fortalecendo os laços afetivos que uniram e nutriram a equipe de desenvolvimento, mesmo com todos os entraves que ocorrem no caminho.

Referências

BARROS, A., 2006. Monografia de Seminários em Sistemas Distribuídos – Uma Análise da Evolução da Coleta de Lixo Distribuída. Disponível em: http://www-di.inf.puc-rio.br/~endler/semGSD/monografias/Alexandra-ColetaLixo.pdf

BITTENCOURT, J. R., OSÓRIO, F. S., 2006. Motores para criação de jogos digitais: gráficos, áudio, interação, rede, inteligência artificial e física.

GAMMA, E., HELM, R., JOHNSON R., VLISSIDES J., 1995. Design Patterns – Elements of Reusable Object-Oriented Software.

HOM, J., 1998. The Usability Methods Toolbox Handbook. Disponível em: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecture notes/UsabilityMethodsToolboxHandbook.pdf

MOITA, F. M. G. S. C., 2009. O mundo virtual e estratégias de estudo: percepções de universitários brasileiros residentes no second life. Revista CoLearn, v.2, p 4-36.

PERUCIA, A. S, BERTHÊM, A. C., BERTSCHINGER, G. L., MENEZES, R. R. C., 2005. Desenvolvimento de jogos eletrônicos. São Paulo: Novatec.

Referências de jogos

CONSPIRAÇÃO DUMONT [2008]. Diário oficial de produção: http://conspiraçãdumont.blogspot.com, disponível em: http://www.abragames.org/JogosBR.html

FULL THROTTLE [1995]. Desenvolvido pela LucasArts Entertainment.

GUITAR HERO [2005..2009]. Desenvolvido pela Harmonix Music Systems e publicado pela RedOctane.

RUNAWAY [2001..2009]. Desenvolvido pela Pendulo Studios.

THE DIG [1995]. Desenvolvido pela LucasArts Entertainment.

THE SECRET OF MONKEY ISLAND [1990]. Desenvolvido pela LucasArts Entertainment.

WHERE IN THE WORLD IS CARMEN SANDIEGO? [1985]. Desenvolvido por pela Brøderbund Software.