

Como Pensar, Implementar e Publicar Jogos Sociais Independentes

Jerry F. Medeiros¹, Paulo H. Abreu¹

¹Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais
{jerrylewisbh, phenriqueabreu}@gmail.com

Abstract. *This tutorial aims to show the independent process of web game's development and publishing to a social network with a view to its game design, implementation, promotion and possible money earning. Approaches, between this and that, important aspects to be taken into consideration during the process of game design for social gaming, installation, configuration and basic features of the OpenSocial using JavaScript, development of simple gameplay with Flash ActionScript3, including advertising, publishing and testing in OrkutSandbox.*

Resumo. *O presente tutorial tem por objetivo demonstrar o processo de desenvolvimento independente e de publicação de um webgame para rede social, tendo em vista seu conceito, implementação, divulgação e possível retorno financeiro. Abordará, portanto, os aspectos importantes a serem levados em consideração durante o processo de game design de jogos sociais, a instalação, configuração e funcionalidades básicas da OpenSocial com JavaScript, desenvolvimento do gameplay com ActionScript 3, inclusão de propagandas, testes e publicação no OrkutSandbox.*

1. Introdução

As redes sociais têm crescido consideravelmente e, com elas, os jogos casuais começam a se multiplicar. Por esse motivo, o mercado brasileiro tem se voltado cada vez mais para o desenvolvimento de jogos para redes sociais, visto que o investimento é bem inferior ao de jogos de prateleira e o retorno financeiro é garantido. Nesse contexto, este tutorial destina-se a estudantes e a profissionais da área de Jogos digitais que estejam interessados em ingressar neste mercado. De maneira simples, o tutorial abordará os seguintes conceitos inerentes ao processo de desenvolvimento e de mercado de jogo sociais:

- Como pensar o Game Design voltado para redes sociais;
- Instalação e configuração da OpenSocial (API livre para criação de aplicativos sociais);

- Funcionalidades básicas da OpenSocial (Obter dados de amigos, postar atualizações no perfil e fazer armazenamento persistente de dados) com JavaScript;
- Desenvolvimento básico em Flash com Actionscript3 (GamePlay);
- Integração dos dados obtidos pela OpenSocial com a aplicação Flash por meio da classe ExternalInterface;
- Como inserir propagandas lucrativas por meio do MochiADS (API livre para inclusão de propagandas em games);
- O que são micro transações e como elas podem ser lucrativas;
- Como publicar e testar aplicações no Orkut SandBox;
- Como conseguir um patrocinador ou Publisher por meio do FlashGameLicense.

A metodologia será voltada para a exposição oral e ilustrada, além de arguições livres. Para tanto, será desenvolvido um jogo similar ao space invaders, no qual os inimigos carregarão bandeiras com fotos obtidas de um perfil do Orkut. O jogo será completamente desenvolvido durante a apresentação, abrangendo todos os tópicos acima listados.

2. Game Design Para Redes Sociais

Para compreender como um jogo para redes sociais deve ser planejado é necessário entender as características do público ao qual ele se destina. Em geral, os usuários de jogos em redes sociais são jogadores casuais. O termo casual refere-se a jogos de baixo custo de produção, com regras simples e pouca duração. Podem exigir bastante habilidade e inteligência do jogador, mas não exigirão tempo e dedicação. Segundo pesquisa divulgada pela Macrovision em 2006:

Em relação à idade:

- 37% entre 35 e 49
- 28% entre 50 e 60
- 71% mulheres
- 29% homens
- 58% não têm filhos e moram com os pais

Em relação ao tempo gasto:

- 37% jogam mais de nove vezes por semana
- 66% dizem que cada partida dura pelo menos uma hora
- 31% gasta mais de 2 horas por partida
- 73% jogam a noite

- 55% jogam nos fins de semana
- 48% jogam em casa

Embora os usuários de jogos em redes sociais praticamente se enquadrem no perfil dos jogadores casuais, existe um fator que os diferencia: a predisposição para interagir com a rede social e usuários a ela conectados. É exatamente por isso que os jogos sociais ganham tantos adeptos: as possibilidades de interação, troca de experiências e criação de conteúdo.

Os jogos sociais precisam ser simples, graficamente atraentes, competitivos, interativos e devem oferecer recompensas, mesmo que o usuário passe muito pouco tempo dentro do ambiente do jogo. As conquistas, chamadas *achievements*, são objetivos paralelos ao objetivo central do jogo, tais como executar uma tarefa em determinado tempo ou convidar certa quantidade de amigos, dentre outras.

O fator mais importante dos jogos sociais é interconectar os usuários, fazer com que eles tenham muitos familiares, vizinhos, empregados dentro do jogo. A quantidade de amigos deve influenciar diretamente na jogabilidade de forma que cada jogador seja instigado a trazer outras pessoas para o jogo.

Outra forma de atrair novos usuários e cativar os atuais é fornecer opção para postar os progresso e conquistas no perfil da rede social, as chamadas atividades.

A rotatividade dos jogadores em redes é alta, principalmente porque a quantidade de jogos que surge todos os dias é enorme. O truque para manter a fidelidade do jogador é oferecer novidades sempre que possível, mesmo que eles tenham que pagar por isso e, nesse caso, entram as micro-transações, que serão detalhadas na seção 8.1.

3. Instalação e Configuração da OpenSocial

Para começar a usar a OpenSocial é necessário ter dois softwares instalados:

- Java SE Development Kit 5.0 ou versão superior, encontrado em:
<http://java.sun.com/javase/downloads/index.jsp>
- Eclipse IDE 3.4.1 ou versão superior encontrado no link:
<http://www.eclipse.org/downloads>

Após a instalação, siga os seguintes passos para criar um projeto OpenSocial:

- Primeiro abra o Eclipse
 - Vá ao menu “Help” e clique em “Install New Software...”
 - Clique em “Add...”
 - Adicione a seguinte URL no campo “Location”:
<http://opensocial-development-environment.googlecode.com/svn/update-site/site.xml>
 - Clique em OK

- Você verá listado OSDE
- Selecione OSDE
- Clique em “Install...”
- Após algum tempo você verá uma caixa de diálogo
- Clique “Next”
- Clique “I accept...” e depois “Finish”
- O Eclipse começará o download e instalação da OSDE
- Quando você ver a caixa de reinicialização do Eclipse clique em “Yes”
- A instalação vai estar completa após o Eclipse reiniciar

Após estes passos, quando criar um novo projeto, você verá disponível um tipo de projeto chamado OSDE.

4. Funcionalidades Básicas da OpenSocial

A OpenSocial é uma plataforma do Google baseada em HTML convencional e Javascript. Uma API (Interface de Programação de Aplicativos) aberta que permite que desenvolvedores criem aplicações para redes sociais. Após gerar o projeto existirão dois arquivos: um XML e um JS (o qual chamaremos de canvas). Todas as funções JavaScript deverão estar contidas no canvas.js.

A OpenSocial possui três áreas principais de recurso:

4.1. Pessoas e relacionamentos

Membros de redes sociais possuem amigos e a OpenSocial disponibiliza os dados e as conexões entre as pessoas e seus amigos.

Exemplo de código:

```
// todos os dados do dono do perfil
var myData;

// todos os dados dos amigos do dono do perfil
var myFriendsData = new Array();
```

Para requisitar os dados do dono do perfil:

```
function getMyData()
{
```

```
//cria um novo container de requisições
var req = opensocial.newDataRequest();

//dispara uma requisição para pegar meus dados e vinculá-los a uma chave que será
chamada de 'viewer'
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.VIEWER), 'viewer');

//envia a requisição. O parâmetro é loadMyData é a função que será executada
quando a requisição for respondida pela OpenSocial
req.send( loadMyData );
}
```

A função de resposta :

```
function loadMyData( data )
{
    //pega todos os dados retornados da chave 'viewer'
    myData = data.get('viewer').getData();
}
```

Agora a requisição dos dados de todos os amigos na rede:

```
function getMyFriends( max )
{
    //cria um novo container de requisições
    var req = opensocial.newDataRequest();

    //cria os parâmetros para a busca dos amigos do viewer
    var paramsFetch = {};
    var idSpec_params = {};
    idSpec_params[opensocial.IdSpec.Field.USER_ID] =
opensocial.IdSpec.PersonId.VIEWER;
    idSpec_params[opensocial.IdSpec.Field.GROUP_ID] = 'FRIENDS';

    //cria uma chave para buscar os dados de uma pessoa ou um grupo de pessoas
    var idSpec = opensocial.newIdSpec(idSpec_params);
}
```

```

    if( max != -1)
        paramsFetch = { "max" : max };

    //insere uma requisição para pegar os dados dos amigos
    req.add(req.newFetchPeopleRequest(idSpec, paramsFetch),
'friends');

    //envia a requisição e retorna os dados na função loadMyFriendsData
    req.send( loadMyFriendsData);
}

```

A função de resposta para os amigos:

```

function loadMyFriendsData( data )
{
    //pega todos os amigos
    var friends = data.get('friends').getData();

    friends.each( function( friend ) {
                                myFriendsData.push( friend );
                            });
}

```

Para obter a quantidade de amigos na rede:

```

function getFriendsLength()
{
    return myFriendsData.length;
}

```

Para obter um dado específico de um amigo específico passa-se o índice do amigo e o dado que se deseja obter:

```

function getFriendData( index, param )
{
    var i = 0;
    var ret;

```

```

    for(friend in myFriendsData)
    {
        if(index == i )
        {
            ret = friend.getField( param );
            break;
        }

        i = i + 1;
    }

    return ret;
}

```

Da mesma forma, para obter os dados do dono do perfil passa-se o nome do campo que se deseja obter:

```

function getData( param )
{

    if(param == opensocial.Person.Field.NAME)
        return myData.getDisplayName();
    else
        if(param == opensocial.Person.Field.ID)
            return myData.getId();

    return myData.getField( param );
}

```

4.2. Persistência

Permite armazenar dados que não serão perdidos quando o usuário fechar a aplicação ou o browser.

Exemplo de código:

```

var myStorageData = new Array(); // todos os dados persistidos do viewer

```

```
var myStorageNameField = new Array("nivel", "xp"); // todos os nomes dos campos criados pelo viewer
```

Para criar ou atualizar o valor de um campo basta passar o nome do campo e o valor da atribuição:

```
function updateFieldData( field, data )
{
    var dataReq = opensocial.newDataRequest();

    dataReq.add(dataReq.newUpdatePersonAppDataRequest( "VIEWER",
field, data));

    dataReq.send(handleUpdateFieldData);
}
```

A função `handleUpdateFieldData` será chamada assim que os dados forem atualizados ou se ocorrer um erro na atualização.

```
function handleUpdateFieldData( data )
{
    if( data.hasError() )
    {
        //tratar o erro
        return;
    }

    //fazer algo
}
```

Para recuperar os dados da armazenados:

```
function requestFieldData( )
{
    var dataReq = opensocial.newDataRequest();

    var fields = myStorageNameField;

    dataReq.add(dataReq.newFetchPersonAppDataRequest( "VIEWER",
fields), "viewer_data");
}
```

```
dataReq.send(handleRequestFieldData);
}
```

A função `handleRequestFieldData` será executada quando os dados forem retornados pela `OpenSocial`.

```
function handleRequestFieldData( data )
{
    var viewerData = data.get("viewer_data");

    if( viewerData.handleError() )
    {
        // tratar o erro

        return;
    }

    var d = viewerData.getData()[ myData.getId() ];

    myStorageData.push( d[myStorageNameField[0]] ); // nivel
    myStorageData.push( d[myStorageNameField[1]] ); // xp
}
```

4.3. Atividades

Os usuários da rede podem informar às outras pessoas o que estão fazendo dentro da aplicação.

Exemplo de código para criar uma atividade:

```
function postActivity( text ) {
    var params = {};
    params[opensocial.Activity.Field.TITLE] = text;
    var activity = opensocial.newActivity(params);
    opensocial.requestCreateActivity(activity,
    opensocial.CreateActivityPriority.HIGH, callback);
}
```

O usuário deverá aceitar postar a atividade no perfil. A seguinte chamada informa se ele o fez ou não:

```
function callback(status) {  
    if (status.hasError())  
    {  
        // tratar erro  
    }  
    else  
    {  
        // sucesso  
    }  
}
```

5. Desenvolvimento de um Jogo Simples com ActionScript 3

Nessa seção veremos como criar um game simples em Flash. Por questões de simplificação não serão usados conceitos de Orientação a Objetos.

Faremos um SpaceInvaders cujo controle da nave será pelo mouse. Incluiremos, para fins de consulta, apenas os trechos principais do código.

Primeiramente, vejamos como movimentar a nave:

```
function moveAround(e:MouseEvent) {  
    //igualar a posição X da nave à posição X do cursor, sem permitir  
    // que a nave ultrapasse as bordas do palco  
    if ( nave_mc.x < stage.stageWidth - nave_mc.width ) {  
        nave_mc.x = mouseX;  
    }  
    if ( nave_mc.x > 0 ) {  
        nave_mc.x = mouseX;  
    }  
}
```

Para que a função funcione é necessário registrar um escutador para o evento de movimentação do Mouse, da seguinte forma:

```
stage.addEventListener( MouseEvent.MOUSE_MOVE, moveAround );
```

A nave atirárá com o clique do Mouse:

```
function shot(e:MouseEvent) {

    var bullet:Bullet = new Bullet();
    bullet.x=nave_mc.respawnPoint_mc.x+nave_mc.x;
    bullet.y=nave_mc.respawnPoint_mc.y+nave_mc.y;

    addChild(bullet);
    bullet.init(mvEnemies);
}
//adiciona um listener para esperar o click de mouse
stage.addEventListener(MouseEvent.CLICK, shot);
```

Os inimigos se movimentarão para um lado e para o outro, baseados em um Timer:

```
function moveEnemies( e:TimerEvent ) {
    if (rightTimes<2&&leftTimes==0) {
        mvEnemies.x += (new Enemy).width;
        rightTimes++;
    } else if ( leftTimes < 2 ) {
        mvEnemies.x -= (new Enemy).width;
        leftTimes++;
    } else if ( downTime > 0 ) {
        mvEnemies.y += (new Enemy).height;

        rightTimes=0;
        leftTimes=0;
        downTime=0;
    } else {
        downTime=1;
        shotEnemy();
    }
}
```

As colisões, por motivo de simplicidade, serão detectadas pela bala:

```
// verifica se a bala encostou em um dos inimigos
function checkCollisions(enemy:MovieClip) {
```

```

        for (var i:int = 0; i<enemy.numChildren; i++) {

            if (this.hitTestObject(enemy.getChildAt(i))) {
                destroyEnemy(enemy.getChildAt(i) as MovieClip);
            }
        }
    }
}

```

```

//verifica ao fim de cada frame se a bala colidiu
function manageCollisions(e:Event) {

    checkCollisions(enemies);

}

```

```

//destroi o inimigo no qual a bala encostou
function destroyEnemy(enemy:MovieClip) {

    enemy.parent.removeChild(enemy);
    destroyBullet();

}

```

```

// destroi a bala
function destroyBullet() {
    this.removeEventListener(Event.ENTER_FRAME, move);
    this.removeEventListener(Event.EXIT_FRAME, manageCollisions);
    MovieClip(this.parent).removeChild(this);
}

```

Infelizmente não é possível reproduzir o código completo do jogo na versão escrita. Esperamos, portanto, que ela sirva como um guia para aqueles que assistiram ao tutorial.

6. Integração do Flash com a OpenSocial

Depois de criadas as funções necessárias no Canvas, conforme descrito na seção 4, é necessário acessá-las de dentro do Flash. Isso pode ser facilmente feito com a classe `ExternalInterface`, a qual permite a comunicação entre o Flash e o browser.

Para que ela se torne disponível é necessário incluí-la no projeto:

```
// Importa a classe ExternalInterface
import flash.external.ExternalInterface;
```

As funções são chamadas da seguinte forma:

```
// Chama a função JavaScript, contida no canvas
ExternalInterface.call("nomeDoMétodo", parametro));
```

O nome do método deve ser passado como primeiro parâmetro, entre aspas e escrito exatamente da mesma forma como foi declarado no canvas (case-sensitive). Em seguida passa-se os parâmetros que a função do Java requer, separados por vírgula, caso existam.

Se o método JavaScript retornar algum parâmetro, ele será enviado ao Flash com o tipo `Object` e deverá ser tratado no próprio Flash. Ex.:

```
// Declara e inicializa um objeto String
var meuRetorno:String = "";

// Atribui o retorno da função nomeDoMetodo à variável meuRetorno
meuRetorno = ExternalInterface.call("nomeDoMetodo", parametro));
```

Desta forma é possível comunicar a aplicação Flash com a OpenSocial.

7. Como Publicar e Testar Aplicações no Orkut SandBox

Para permitir que os desenvolvedores testem suas aplicações no ambiente do Orkut a Google criou uma ferramenta chamada OrkutSandbox. A interface é extremamente semelhante à do orkut e todos os dados dos seus amigos estarão disponíveis para uso da aplicação.

Inscriver-se no SandBox é simples, basta ter uma conta no Orkut tradicional e inscrever-se em: <http://sandbox.orkut.com/SandboxSignup.aspx>.

Depois de inscrito, acesse a opção Apps -> Editar, na lateral esquerda da tela. Haverá a opção "Se preferir, adicione um aplicativo diretamente pelo URL:". Basta colocar a url na qual se encontra o XML da sua aplicação. É altamente recomendável utilizar a versão antiga do Orkut.

Quando a aplicação estiver concluída e devidamente testada o envio deve ser feito no Orkut e não no sandbox. Basta acessar a opção Aplicativo -> Enviar seu aplicativo.

A aplicação passará por uma avaliação que dura em média uma semana. Após aprovada, a Google enviará um código que certifica a propriedade da aplicação, a qual deverá ser incluída no XML. Feito isso, a aplicação já estará disponível para os usuários do Orkut.

8. Ganhando Dinheiro com o seu jogo

Depois de tanto trabalho para pensar e desenvolver um jogo legal é natural que se espere uma receita proporcional ao esforço empregado. Existem diversas formas de monetizar um jogo. Aqui, abordaremos três delas: inclusão de propagandas, venda de "bens-virtuais" e licenciamento para patrocinadores.

8.1. Micro-Transações

Microtransações são operações financeiras que envolvem uma quantia muito pequena de dinheiro. Para o Paypal - uma empresa que permite a transferência de dinheiro entre indivíduos ou negociantes - transferências de até doze dólares americanos são consideradas microtransações.

Nos jogos sociais as microtransações servem para a venda de "bens-virtuais" no ambiente do jogo e incluem, dentre outras coisas, customização de avatares e cenários, habilidades, power-ups, armas, de acordo com as características do jogo em si.

Embora muitos usuários paguem apenas pela estética, a maioria deles espera obter vantagens daquilo que foi comprado. Portanto, é importante que alguns dos "bens-virtuais" influenciem diretamente na jogabilidade. Uma estratégia interessante que tem sido amplamente utilizada é a criação de duas "moedas" distintas, sendo possível adquirir uma delas apenas por microtransações.

8.2. MochiAds

Outra forma de monetizar jogos na web é a inclusão de propagandas. O Mochiads é uma ferramenta gratuita para inclusão de propagandas, voltava especificamente para jogos web, os quais usem Flash Player como plataforma

Os anúncios podem se adicionados antes do início do jogo - mostrados durante o tempo de carregamento, que às vezes é estendido para permitir o anúncio inteiro seja visualizado – ou entre as fases. O ganho médio é de 0.5 centavos de dólar a cada mil impressões. À primeira vista parece pouco, mas, se o jogo tiver um milhão de acessos mensais, o ganho aproximado é de 500 (quinhentos) dólares.

Incluir anúncios com a MochiAds é simples, basta cadastrar-se gratuitamente no site: <http://en.mochimedia.com/developers/ads.html>, submeter sua aplicação para avaliação, baixar a API e incluir uma única linha de código, gerada exclusivamente para a sua aplicação.

8.3. Flash Game License

O FlashGameLicense é uma espécie de corretora de jogos feitos em Flash que pode ser utilizada para que os publishers tenham acessos aos seus jogos. Se eles gostarem, farão ofertas de patrocínio.

Para que seu jogo esteja disponível para avaliação dos Publishers basta cadastrar-se em <http://www.flashgamelicense.com/>, submeter e preencher algumas informações simples, como a descrição do gênero, a porcentagem e previsão de conclusão do jogo. Os patrocinadores em potencial podem ver e jogar todos os jogos no banco de dados, ou procurar por gênero ou tipo de licenciamento. Quando um patrocinador encontra um jogo que se adapta às suas necessidades fazem, então, uma oferta. Os desenvolvedores e patrocinadores podem negociar o preço e o tipo de licença.

O interesse dos patrocinadores é o tráfego que um jogo pode trazer para o site dele. Se um patrocinador oferece ao desenvolvedor X dólares por um jogo, é porque espera que o jogo renda um valor superior a X.

Existem basicamente dois modelos de negócio referentes ao licenciamento de jogos para patrocinadores:

Licença exclusiva: O patrocinador é único e pagará para que seu jogo carregue a marca independente de onde esteja hospedado. Geralmente não é permitido incluir outras propagandas no jogo e o patrocinador compra o direito de distribuir o jogo de forma gratuita para qualquer site. O desenvolvedor, no entanto, detém a autoria e pode participar de concursos.

Licença primária: É exclusiva apenas por um tempo determinado em contrato. Após esse período o patrocinador primário não é mais exclusivo. Ele paga ao desenvolvedor para exibir sua marca no jogo. No entanto, o desenvolvedor não perde o direito de produzir versões do jogo para outros sites, com outras propagandas e de vender licença secundárias, terciárias, etc.. A licença secundária custa menos que a primária, a terciária menos que a secundária e assim por diante.

A quantia envolvida no licenciamento de um jogo varia de acordo com vários fatores, dentre os quais está o nível de exclusividade para o patrocinador e do interesse que ele pode despertar no público.

Referências

Flash Game License. “What is a Sponsorship” ,
http://www.flashgamelicense.com/view_library.php?page=what-is-a-sponsorshipPress

PayPal Developer Central . “Micropayments”,
https://www.paypalobjects.com/IntegrationCenter/ic_micropayments.html

Press Release. (2006) “Survey: Casual Gamers Playing 20 Hours a Week”,
<http://news.teamxbox.com/xbox/11263/Survey-Casual-Gamers-Playing-20-Hours-a-Week>, June.

Wikipedia . “OpenSocial”,
<http://en.wikipedia.org/wiki/OpenSocial>