

DENILSON BONATTI

Desenvolvimento de  
**JOGOS** em **HTML5**



- METODOLOGIA PASSO A PASSO
- DESENVOLVA JOGOS PARA WEB,  
TABLETS, CELULARES E FACEBOOK





DENILSON BONATTI

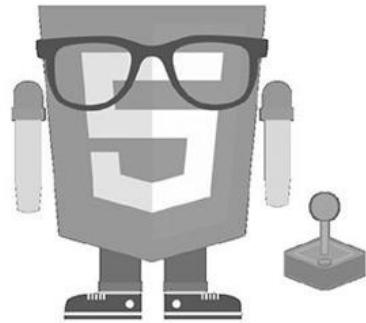
Desenvolvimento de  
**JOGOS** em **HTML5**



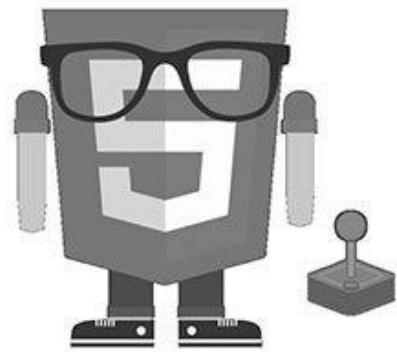
- METODOLOGIA PASSO A PASSO
- DESENVOLVA JOGOS PARA WEB,  
TABLETS, CELULARES E FACEBOOK



Desenvolvimento de  
**JOGOS**<sub>em</sub>  
**HTML5**



DENILSON BONATTI



Desenvolvimento de

# JOGOS em HTML5



Copyright© 2014 por Brasport Livros e Multimídia Ltda.

Todos os direitos reservados. Nenhuma parte deste livro poderá ser reproduzida, sob qualquer meio, especialmente em fotocópia (xerox), sem a permissão, por escrito, da Editora.

Para uma melhor visualização deste e-book sugerimos que mantenha seu software constantemente atualizado.

Editor: Sergio Martins de Oliveira

Diretora Editorial: Rosa Maria Oliveira de Queiroz

Assistente de Produção: Marina dos Anjos Martins de Oliveira

Revisão de Texto: Maria Helena A. M. Oliveira

Eletrônica: SBNigri Artes e Textos Ltda.

Capa: Paulo Vermelho

Produção de e-pub: SBNigri Artes e Textos Ltda.

Técnica e muita atenção foram empregadas na produção deste livro. Porém, erros de digitação e/ou impressão podem ocorrer. Qualquer dúvida, inclusive de conceito, solicitamos enviar mensagem para [brasport@brasport.com.br](mailto:brasport@brasport.com.br), para que nossa equipe, juntamente com o autor, possa esclarecer. A Brasport e o(s) autor(es) não assumem qualquer responsabilidade por eventuais danos ou perdas a pessoas ou bens, originados do uso deste livro.

ISBN Digital: 978-85-7452-701-7

**BRASPORT Livros e Multimídia Ltda.**

Rua Pardal Mallet, 23 – Tijuca

20270-280 Rio de Janeiro-RJ

Tels. Fax: (21) 2568.1415/2568.1507

**e-mails:**

[marketing@brasport.com.br](mailto:marketing@brasport.com.br)

[vendas@brasport.com.br](mailto:vendas@brasport.com.br)

[editorial@brasport.com.br](mailto:editorial@brasport.com.br)

**site: [www.brasport.com.br](http://www.brasport.com.br)**

**Filial**

Av. Paulista, 807 – conj. 915

01311-100 – São Paulo-SP

Tel. Fax (11): 3287.1752

e-mail: [filialsp@brasport.com.br](mailto:filialsp@brasport.com.br)

*À minha esposa Mariana, aos meus pais*

*Maria Cristina e Antônio e aos meus*

*irmãos Rodrigo, Débora, Rogério e Daniel.*

## **Agradecimentos**

A Daniel Bonatti, por testar e analisar os códigos, e a Ralmer Lorenzato, pelo desenvolvimento das imagens utilizadas nos jogos.

## **Apresentação**

O que você irá encontrar neste livro é o resultado da experiência de alguns anos criando material didático sobre desenvolvimento de jogos para diversas instituições de ensino do Brasil. Este livro é destinado a leigos no assunto e àqueles que desejam migrar o desenvolvimento de jogos de outra plataforma para o HTML5.

Caso você já tenha alguma experiência no desenvolvimento de jogos ou em programação, poderá notar a quantidade de imagens explicativas da posição correta e a sintaxe dos códigos, mas lembre-se de que este livro é destinado também para leigos e é pensando neles que as imagens foram inseridas.

Neste livro serão desenvolvidos dois jogos do início ao fim utilizando uma didática “passo a passo”. Serão também mencionadas as formas de publicação dos jogos desenvolvidos em dispositivos móveis, como aplicativo do Facebook e, é claro, na forma de browser game.

## **Sobre o Autor**

Denilson Bonatti é formado em Web Design e possui MBA em Desenvolvimento de Marcas e Marketing, ambos pelo Centro Universitário de Araraquara (UNIARA).

Possui experiência de mais de dez anos ministrando aulas em diversos centros de treinamento. Desenvolveu cursos e materiais didáticos para várias franquias de cursos profissionalizantes e cursos à distância (EAD), entre eles On Byte Formação Profissional ([www.onbyte.info](http://www.onbyte.info)), Remington Educação e Profissão ([www.remington.com.br](http://www.remington.com.br)), Treinaweb ([www.treinaweb.com.br](http://www.treinaweb.com.br)), Cursos 24 horas ([www.cursos24horas.com.br](http://www.cursos24horas.com.br)), Portal Luciano Augusto ([www.lucianoaugusto.com.br](http://www.lucianoaugusto.com.br)) e portal Denilson Bonatti ([www.denilsonbonatti.com.br](http://www.denilsonbonatti.com.br)).

Autor do livro “Desenvolvimento de Sites Dinâmicos com Dreamweaver CC”, também publicado pela Brasport.

### **Fale com o autor:**

Em caso de alguma dúvida, sugestão ou crítica sobre o conteúdo do livro, entre em contato pelo e-mail [contato@denilsonbonatti.com.br](mailto: contato@denilsonbonatti.com.br).

# Sumário

[Introdução](#)

[Capítulo 1 – De quais programas eu preciso para desenvolver um jogo em HTML5?](#)

[Aplicativos](#)

[Notepad++](#)

[Google Chrome](#)

[XAMPP](#)

[Filezilla](#)

[Frameworks](#)

[jQuery](#)

[Collision](#)

[Hammer](#)

[Instalando os programas necessários](#)

[Instalando o Notepad++](#)

[Instalando o Google Chrome](#)

[Instalando o aplicativo XAMPP](#)

[Instalando o Filezilla](#)

[Capítulo 2 – Como utilizar o HTML5 no desenvolvimento de jogos?](#)

[De que forma iremos utilizar o HTML5 no desenvolvimento dos jogos?](#)

[Conhecendo o primeiro jogo](#)

[Chega de papo e vamos ao código](#)

[Capítulo 3 – Arquivo CSS](#)

[Criando um arquivo CSS](#)

[Capítulo 4 – Arquivo JavaScript](#)

[Capítulo 5 – Criando animações utilizando o CSS3](#)

[Capítulo 6 – Game Loop](#)

[Não está funcionando, e agora?](#)

[Capítulo 7 – Detectando teclas pressionadas](#)

[Capítulo 8 – Limitando a movimentação do jogador e criando uma animação pelo JavaScript](#)

[Animando a div inimigo1](#)

[Capítulo 9 – Criando a movimentação das demais divs do jogo](#)

[Capítulo 10 – Criando a função disparo](#)

[Capítulo 11 – Colisões parte I](#)

[Capítulo 12 – Colisões parte II](#)

[Capítulo 13 – Colisões parte III](#)

[Capítulo 14 – Pontuação](#)

[Capítulo 15 – Criando uma barra de energia](#)

[Capítulo 16 – Aumentando a dificuldade do jogo](#)

[Capítulo 17 – Som](#)

[Capítulo 18 – Game over](#)

[Capítulo 19 – Reiniciando o jogo](#)

[Capítulo 20 – Criando uma versão do jogo para dispositivos móveis](#)

[Utilizando uma biblioteca para detectar toques](#)

[Capítulo 21 – Facebook game](#)

[Criando uma conta de desenvolvedor no Facebook](#)

[Criando um novo aplicativo](#)

[O que é um aplicativo para Facebook?](#)

[Utilizando um servidor local](#)

[Executando o jogo no servidor local](#)

[Capítulo 22 – Criando uma conexão com o Facebook](#)

[Configurando JavaScript SDK](#)

[Open Graph \(Graph API\)](#)

[Método FB.getLoginStatus](#)

[Caixa de diálogo OAuth](#)

[Capítulo 23 – Exibindo informações do jogador](#)

[Open Graph](#)

[Capítulo 24 – Facebook Query Language \(FQL\)](#)

[Graph API Explorer](#)

[Capítulo 25 – Compartilhando a pontuação](#)

[Utilizando o método FB.ui](#)

[Plugins sociais](#)

[Capítulo 26 – Hospedando o jogo em um servidor web](#)

[Utilizando o Filezilla](#)

[Atualizando as configurações do aplicativo no Facebook](#)

[Capítulo 27 – Canvas game](#)

[Utilizando uma imagem mapeada](#)

[Capítulo 28 – Criando uma animação com imagens mapeadas](#)

[Capítulo 29 – Criando a movimentação do tanque](#)

[O que será executado na função eventoClick?](#)

[Exibindo o inimigo e a bandeira](#)

[Capítulo 30 – Inteligência artificial](#)

[Capítulo 31 – Mensagens de início e fim de jogo](#)

## Capítulo 32 – Utilizando o PhoneGap

### Conclusão

# Introdução

Todos os dias milhões de usuários gastam minutos e até horas em jogos na internet, e é possível até dizer que uma grande parte desses usuários gasta mais tempo jogando do que checando e-mails ou navegando em comunidades sociais como o Facebook.

O universo do jogo eletrônico é muito grande e está em plena expansão. Jogos eletrônicos são uma forma de entretenimento moderna que a cada dia atrai mais pessoas estimuladas por sua interatividade e poder de imersão. Grande parte deste crescimento se deve aos chamados jogadores casuais, que não se encaixam no estereótipo de um jogador de videogame que passa horas e horas dedicado ao seu jogo preferido. Os jogadores casuais são aqueles que preferem jogos simples e rápidos que podem ser jogados em qualquer local, do seu celular ou online pelo browser de navegação.

Uma das grandes dificuldades encontradas pelos desenvolvedores de jogos é encontrar uma plataforma de desenvolvimento compatível com os diversos dispositivos móveis disponíveis no mercado e também com os diferentes navegadores (browsers). Utilizando o HTML5 juntamente com as folhas de estilos CSS3 e o JavaScript, é possível desenvolver jogos de uma forma rápida, compatível com diversas plataformas, incluindo dispositivos móveis.

## **Capítulo 1**

### **De quais programas eu preciso para desenvolver um jogo em HTML5?**

Para a criação e edição dos códigos HTML5/CSS3/JavaScript não é necessário instalar nenhum aplicativo específico – pode-se até utilizar o bloco de notas do Windows –, mas para um desenvolvimento mais preciso, e para que testes possam ser realizados de maneira mais confiável, utilizaremos os aplicativos listados a seguir.

## **Aplicativos**

## **NotePad++**

O NotePad++ é um editor de texto e códigos-fonte que suporta as mais diversas linguagens de programação. Possui muitas características favoráveis para facilitar o trabalho, como autocompletar, sistema de busca e substituição, interface funcional e navegação por abas.

## **Google Chrome**

O Google Chrome é um dos navegadores mais populares da internet e tem como principal característica a sua velocidade de navegação, sendo totalmente compatível com os principais recursos do CSS3. Utilizaremos o Google Chrome para realizar testes no desenvolvimento dos jogos.

## XAMPP

O aplicativo XAMPP é um programa do tipo WAMP, que agrega várias tecnologias, fornecendo ferramentas que facilitam a criação de websites. A instalação de um WAMP normalmente fornece os seguintes recursos:

- **Banco de dados MySQL:** é um sistema de gerenciamento de banco de dados utilizado em grande escala na internet que usa a linguagem SQL.
- **Servidor Apache:** é o servidor web mais utilizado na internet. Toda página de conteúdo dinâmico necessita de um servidor para que as informações solicitadas pelo código PHP obtenham resposta.

Antes de disponibilizar o jogo via Facebook, enquanto ele estiver em desenvolvimento, realizaremos testes em um servidor web local e o XAMPP será o aplicativo que utilizaremos para a criação do servidor local necessário para os testes.

## **Filezilla**

O aplicativo Filezilla é um cliente FTP que usaremos para realizar o upload dos arquivos do jogo para o servidor web.

## Frameworks

Além dos aplicativos citados, iremos utilizar dois frameworks no desenvolvimento dos jogos. Um framework é um conjunto de classes utilizadas para resolver problemas comuns em diversos tipos de softwares. Frameworks facilitam o desenvolvimento de websites e programas, pois apresentam uma série de funcionalidades prontas para o uso, necessitando apenas de pequenas configurações. Dentre as diversas bibliotecas disponíveis na internet, neste projeto iremos utilizar o jQuery e o Collision.

## **jQuery**

jQuery é uma biblioteca JavaScript utilizada para simplificar a manipulação de eventos, animação e interações em uma página web. Utilizaremos essa biblioteca para simplificar os códigos utilizados nos jogos. O framework jQuery pode ser baixado pelo link [www.jquery.com](http://www.jquery.com), mas não se preocupe, mais à frente será indicado o link onde você poderá baixar todos os arquivos que serão utilizados nos jogos, entre eles os arquivos dos frameworks.

## **Collision**

O framework Collision é uma extensão da biblioteca jQuery utilizada para detectar a colisão entre objetos. Utilizaremos essa biblioteca para facilitar a detecção das colisões do jogo. O Collision pode ser baixado pelo link <http://sourceforge.net/projects/jquerycollision/>, mas não se preocupe, mais à frente será indicado o link onde você poderá baixar todos os arquivos que serão utilizados nos jogos.

## **Hammer**

Hammer é uma biblioteca JavaScript utilizada para identificar gestos multi-touch em dispositivos que suportam essa tecnologia, como tablets e celulares. Os gestos suportados pelo Hammer são: toque, toque duplo, arrastar e transformar (pinça para zoom) etc. Usaremos essa biblioteca para criar uma versão do jogo para dispositivos móveis.

## **Instalando os programas necessários**

Vamos iniciar baixando os arquivos do curso e instalando o Notepad++.

## **Instalando o Notepad++**

1. Abra o seu browser de navegação padrão. Neste exemplo estamos utilizando o Mozilla Firefox (Figura 1).

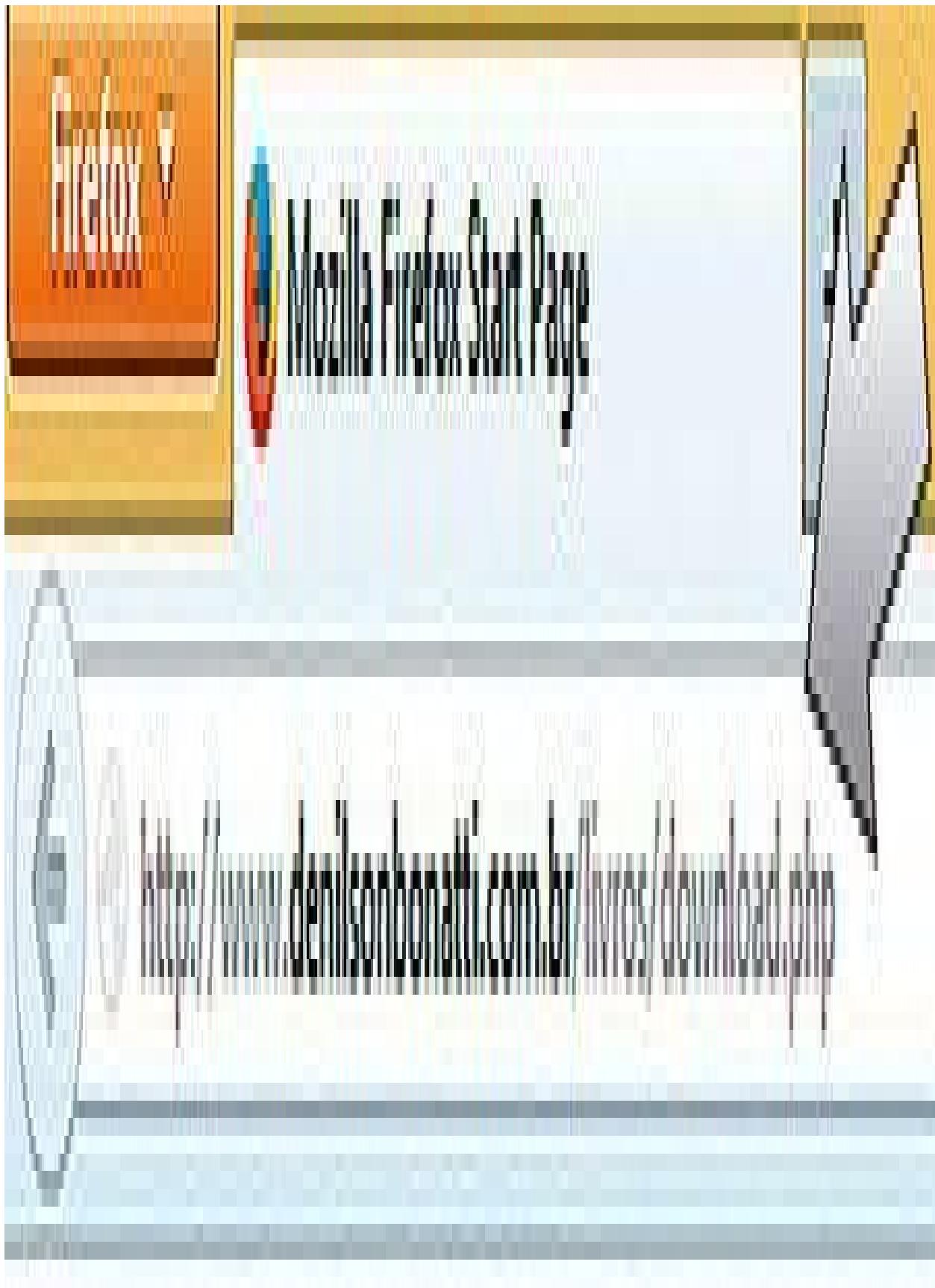


## **Figura 1**

O próximo passo é realizar o download dos arquivos que serão utilizados no desenvolvimento dos jogos, como imagens, sons e códigos.

2. Digite na barra de endereço a seguinte URL:

<http://www.denilsonbonatti.com.br/livros/download.php> (Figura 2).



## **Figura 2**

3. Realize o download do arquivo e descompacte o seu conteúdo. Dê um clique duplo na pasta Programas (Figura 3).

Cedigés

logol

log2

Programas



### **Figura 3**

4. Dê um clique duplo no arquivo npp.6.5.1.Installer.exe (Figura 4).



**npp.6.5.1h**  
**steller**

**xampp-win**

**32.1.8.2.2-**  
**VC9-install**

**er**

#### **Figura 4**

5. Quando a instalação for iniciada, selecione o idioma “Português Brasileiro” e clique no botão “OK” (Figura 5).

# Installer Language



Please select a language.

Portuguese Brasileiro

OK

Cancel

## **Figura 5**

6. Na tela seguinte clique no botão “Próximo”.
7. Nesta tela será exibida a licença de uso do produto. Clique em “Eu concordo” para prosseguir.
8. Na próxima tela será exibido o local de instalação do Notepad++. Clique em “Próximo” para prosseguir com a instalação.
9. Na próxima tela temos as opções de componentes que podemos instalar juntamente com o Notepad++, dentre eles o autocompletar. Deixe as opções padrão de instalação e clique no botão “Próximo” (Figura 6).



## Instalação da Notepad++ v6.5.1



### Escolher Componentes

Escolha quais funções da Notepad++ v6.5.1 você quer instalar.

Marque os componentes que você quer instalar e desmarque os componentes que você não quer instalar. Clique em Próximo para continuar.

Selezione o tipo de instalação: **Personalizado**

Ou, selecione os componentes opcionais que você deseja instalar:

- Context Menu Entry
- Auto-completion Files
- Plugins
- Localization
- Themes
- As default html viewer
- Auto-Updater
- User Manual

#### Descrição

Posicione seu mouse sobre um componente para ver sua descrição.

Espaço requerido: 13.0MB

Don HO:



< Voltar

Próximo >

Cancelar

## **Figura 6**

10. Na próxima tela temos a opção de criar um ícone do Notepad++ no desktop (área de trabalho). Selecione a opção “Create Shortcut on Desktop” e em seguida clique em “Instalar” (Figura 7).



Instalação da Notepad++ v6.5.1



## Escolher Componentes

Escolha quais funções da Notepad++ v6.5.1 você quer instalar.

Don't use %APPDATA%

Enable this option to make Notepad++ load/write the configuration files from/to its install directory. Check it if you use Notepad++ in an USB device.

Allow plugins to be loaded from %APPDATA%\notepad++\plugins

It could cause a security issue. Turn it on if you know what you are doing.

Create Shortcut on Desktop

Use the old, obsolete and monstrous icon

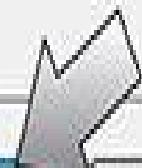
I won't blame you if you want to get the old icon back :)

Don HO -

< Voltar

Instalar

Cancelar



## **Figura 7**

11. Aguarde o término da instalação. Na última etapa de instalação desabilite a opção “Executar Notepad++” e clique no botão “Terminar” (Figura 8).



Instalação da Notepad++ v6.5.1



## Concluindo a instalação da Notepad++ v6.5.1

A Notepad++ v6.5.1 foi instalada no seu computador.

Clique em **Terminar** para fechar este assistente.

Executar Notepad++ v6.5.1

< Voltar

**Terminar**

Cancelar

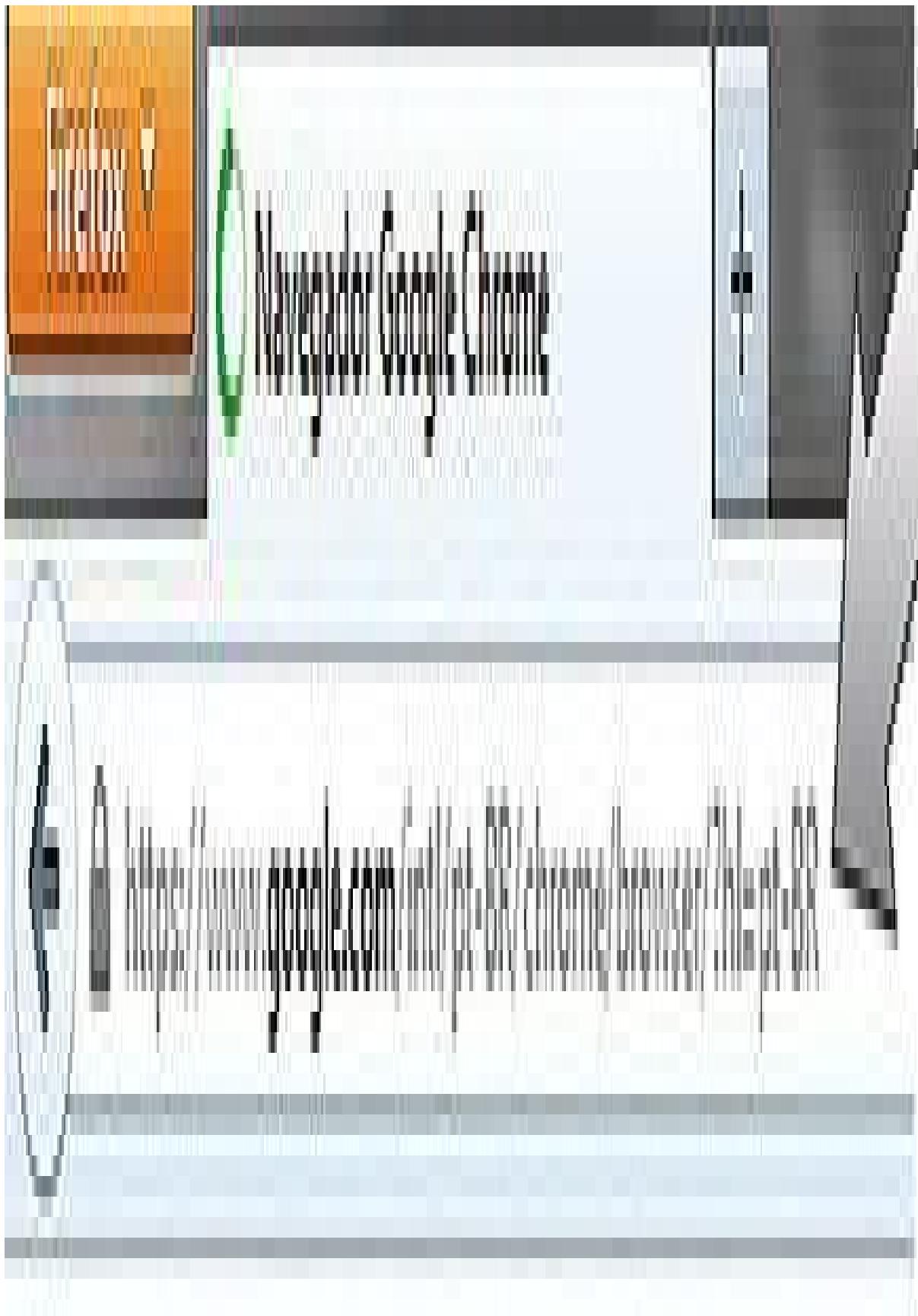


## **Figura 8**

Com o Notepad++ instalado, o próximo passo é instalar o Google Chrome.

## **Instalando o Google Chrome**

1. Na barra de endereços do Mozilla Firefox digite a seguinte URL:  
<https://www.google.com/intl/pt-BR/chrome/browser/?hl=pt-BR> e pressione a tecla Enter (Figura 9).



## **Figura 9**

2. Quando a página for exibida, clique no botão “Fazer o download do Google Chrome” (Figura 10).



Particulars of the  
Art of War

## **Figura 10**

3. Quando a janela com os termos de uso for exibida, clique no botão “Aceitar e instalar” (Figura 11).



## **Figura 11**

4. Aguarde a realização do download e da instalação.

## **Instalando o aplicativo XAMPP**

1. Abra a pasta Programas e dê um clique duplo no arquivo xampp-win32-1.8.2-2-VC9-installer.exe (Figura 12).



app651.b

stl



app651.b

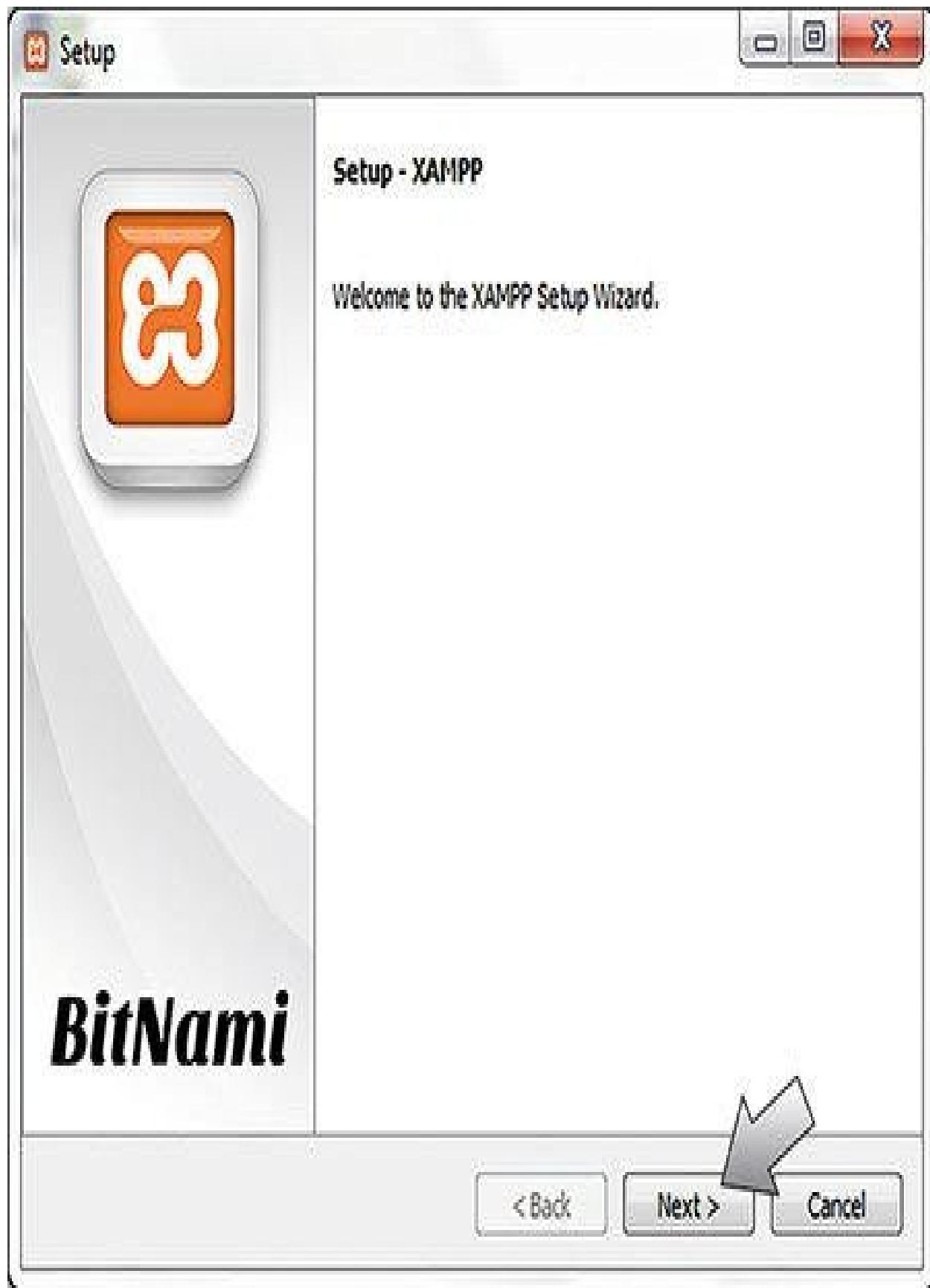
stl

VC9-install



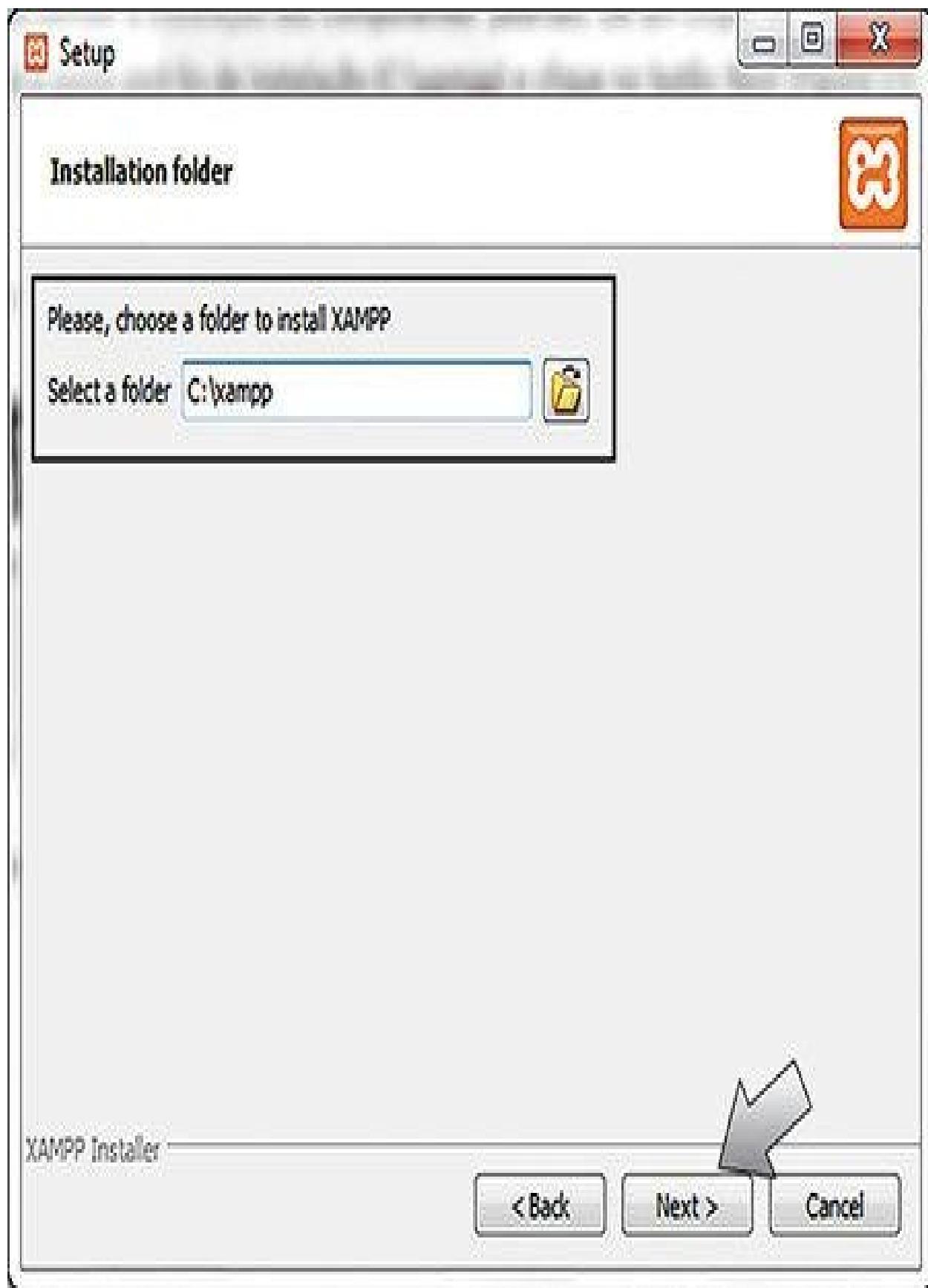
## **Figura 12**

2. Ao iniciar a instalação, uma tela de boas-vindas será exibida. Dê um clique no botão “Next” (Figura 13).



### **Figura 13**

3. Usaremos a instalação dos componentes padrões. Dê um clique no botão “Next”.
4. Deixe a pasta padrão de instalação (C:\xampp) e clique no botão “Next” (Figura 14).



### **Figura 14**

5. Desabilite a opção “Learn more about BitNami for Xampp” e clique no botão “Next”.
6. Clique novamente no botão “Next” para dar início à instalação.

## Instalando o Filezilla

1. Dê um clique no arquivo de instalação FileZilla\_3.7.3\_win32-setup.exe.
2. Na primeira janela de instalação será exibido um contrato de utilização do aplicativo. Dê um clique no botão “I Agree”.
3. Selecione a opção “Anyone who uses this computer (all users)” para que todos os usuários do sistema tenham acesso ao aplicativo. Clique no botão “Next”.
4. Deixe todos os componentes padrão selecionados e clique no botão “Next”.
5. Deixe a pasta padrão de instalação selecionada e clique no botão “Next”.
6. Para finalizar clique no botão “Install” e aguarde o final da instalação.

## **Capítulo 2**

# **Como utilizar o HTML5 no desenvolvimento de jogos?**

No desenvolvimento de jogos ou qualquer aplicativo utilizando o HTML5, alguns mitos de desenvolvimento devem ser explicados.

A estrutura de desenvolvimento de um aplicativo HTML5 segue as seguintes características:

-

HTML5	Marcação
CSS3	Formatação
JavaScript	Comportamento

## ■

### O que isso quer dizer?

Ao se criar um jogo ou aplicativo utilizando HTML5, o HTML5 somente será responsável pela exibição dos elementos no browser, como imagens, textos, execução de sons, exibição de vídeos etc.

Para realizar a formação desses objetos utilizaremos o CSS3, e para criar comportamentos e funções, ou seja, a programação do jogo, usaremos o JavaScript. O HTML5 não apresenta métodos e funções para detecção de comportamentos, o que deve ser feito no código JavaScript. O que o HTML5 trouxe de novo são as novas possibilidades para que o JavaScript controle os elementos criados no código. O HTML5 será o novo padrão da linguagem de marcação de hipertextos (HTML). O HTML5 ainda está em fase de desenvolvimento, mas a maioria de seus novos recursos já pode ser utilizada em browsers modernos como o Google Chrome, Mozilla Firefox, Safari etc.

### O que há de novo no HTML5 e quais as vantagens de sua utilização:

- Elemento <canvas> para desenho 2D e 3D.
- Elementos <video> e <audio> para reprodução de mídias.
- Suporte para armazenamento local.
- Novas tags para conteúdos específicos como: <article>, <header>, <foot>, <section> etc.
- Novos controles de formulários, como calendários, data, hora, pesquisa etc.

Para uma referência completa dos novos itens do HTML5 utilize o link a seguir:

[http://www.w3schools.com/html/html5\\_new\\_elements.asp](http://www.w3schools.com/html/html5_new_elements.asp)

## **De que forma iremos utilizar o HTML5 no desenvolvimento dos jogos?**

Neste livro iremos desenvolver dois jogos. No primeiro jogo, vamos utilizar o Modelo de Objetos do Documento (DOM, na sigla em inglês), e no segundo, o recurso canvas do HTML5.

Utilizando o modelo DOM, utilizaremos o JavaScript para dar interatividade e movimentação ao jogo e o CSS3 para criar as animações.

No segundo jogo, vamos utilizar o canvas para renderizar e exibir as imagens no browser e o JavaScript para criar as animações e a interatividade.

## **Conhecendo o primeiro jogo**

No primeiro jogo, que terá o nome de Resgate, o usuário irá controlar um helicóptero, cuja função será resgatar os seus amigos e eliminar os helicópteros inimigos (Figura 15).



## **Figura 15**

Você pode jogá-lo acessando o link a seguir:

[www.denilsonbonatti.com.br/livros/jogo1](http://www.denilsonbonatti.com.br/livros/jogo1)

## **Chega de papo e vamos ao código**

Inicialmente vamos copiar os arquivos necessários para o desenvolvimento do jogo.

1. Crie uma pasta com o nome de Resgate.

O próximo passo é copiar o conteúdo da pasta jogo1 dos arquivos baixados em <http://www.denilsonbonatti.com.br/livros/download.php> para a pasta recém-criada.

2. Abra a pasta jogo1.

Observe que você encontrará as seguintes pastas (Figura 16).



fontes



imgs



js



OutrasImagens



songs

## **Figura 16**

3. Copie as pastas indicadas na Figura 16 para a pasta Resgate.
4. Execute o aplicativo Notepad++.

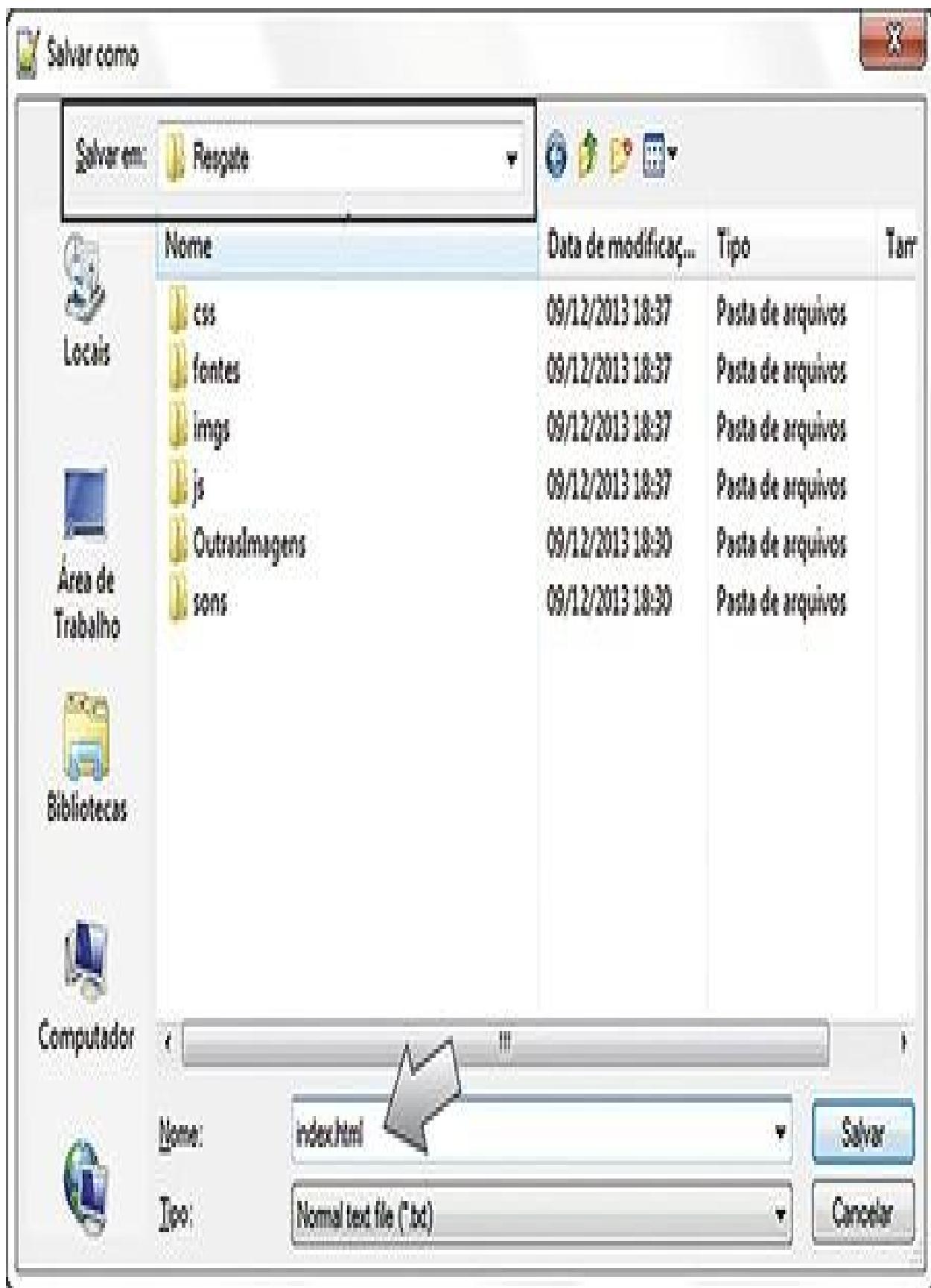
Antes de digitar o código, vamos salvar este arquivo dentro da pasta Resgate.

5. Dê um clique no menu Arquivo e selecione a opção “Salvar como” (Figura 17).



### **Figura 17**

6. Salve o arquivo com o nome de index.html dentro da pasta Resgate, como indicado na Figura 18.



## Figura 18

A seguir vamos digitar o código de uma estrutura básica de um arquivo HTML5.

7. Digite o código a seguir:

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Resgate</title>

</head>

<body>

<div id="container"> <!-- div container !-->

<div id="fundoGame"> <!-- div fundoGame !-->

<div id="inicio"><h1> Resgate </h1> <!-- div in cio !-->

<p>Utilize a tecla W para movimentar o helic ptero para cima, a tecla S para
movimentar o helic ptero para baixo e a tecla D para atirar.</p> <p> Clique aqui
```

para iniciar!! </p>

```
</div> <!-- Fim da div início !-->  
</div> <!-- Fim da div fundoGame !-->  
</div> <!-- Fim da div container !-->  
</body>  
</html>
```

■ Respeite sempre os caracteres maiúsculos e minúsculos na digitação do código.

8. Utilize as teclas Ctrl + S para salvar as alterações no arquivo.

Os códigos apresentados neste livro estão salvos dentro da pasta código dos arquivos.

■

Observe que apenas criamos a estrutura de divs que serão utilizadas no jogo, mas ainda não indicamos a posição, o tamanho e a formatação de cada uma das divs criadas. Isso será realizado no próximo capítulo, com a criação do arquivo CSS.

## **Capítulo 3**

### **Arquivo CSS**

As folhas de estilo CSS servem para promover o acabamento visual das páginas web. Elas podem ser compartilhadas entre várias páginas, permitindo assim uma padronização visual muito simplificada e lógica. Utilizaremos os códigos CSS inicialmente para posicionar e indicar o tamanho de cada div e indicar as demais formatações dos elementos que serão utilizados no jogo.

## Criando um arquivo CSS

CSS é a abreviação do termo em inglês Cascading Style Sheet, que é traduzido para o português como folhas de estilo em cascata. O CSS tem por finalidade indicar como uma determinada marcação em HTML deve ser exibida, como, por exemplo, cor, tamanho, posição etc.

Segundo a W3C (World Wide Web Consortium), que é um consórcio internacional de pessoas, empresas e órgãos governamentais que visa desenvolver padrões para a criação e a interpretação de conteúdos para a web, CSS pode ser definido da seguinte maneira:

“CSS é um mecanismo simples para adicionar estilos, por exemplo, fontes, cores e espaçamentos, aos documentos web”.

Na Figura 19 temos uma regra básica da anatomia de um código CSS, que deve ser composta pelo seletor e a declaração que será atribuída a esse seletor, composta por propriedade e valor:

# Regra CSS

```
seletor { propriedade: valor };
```

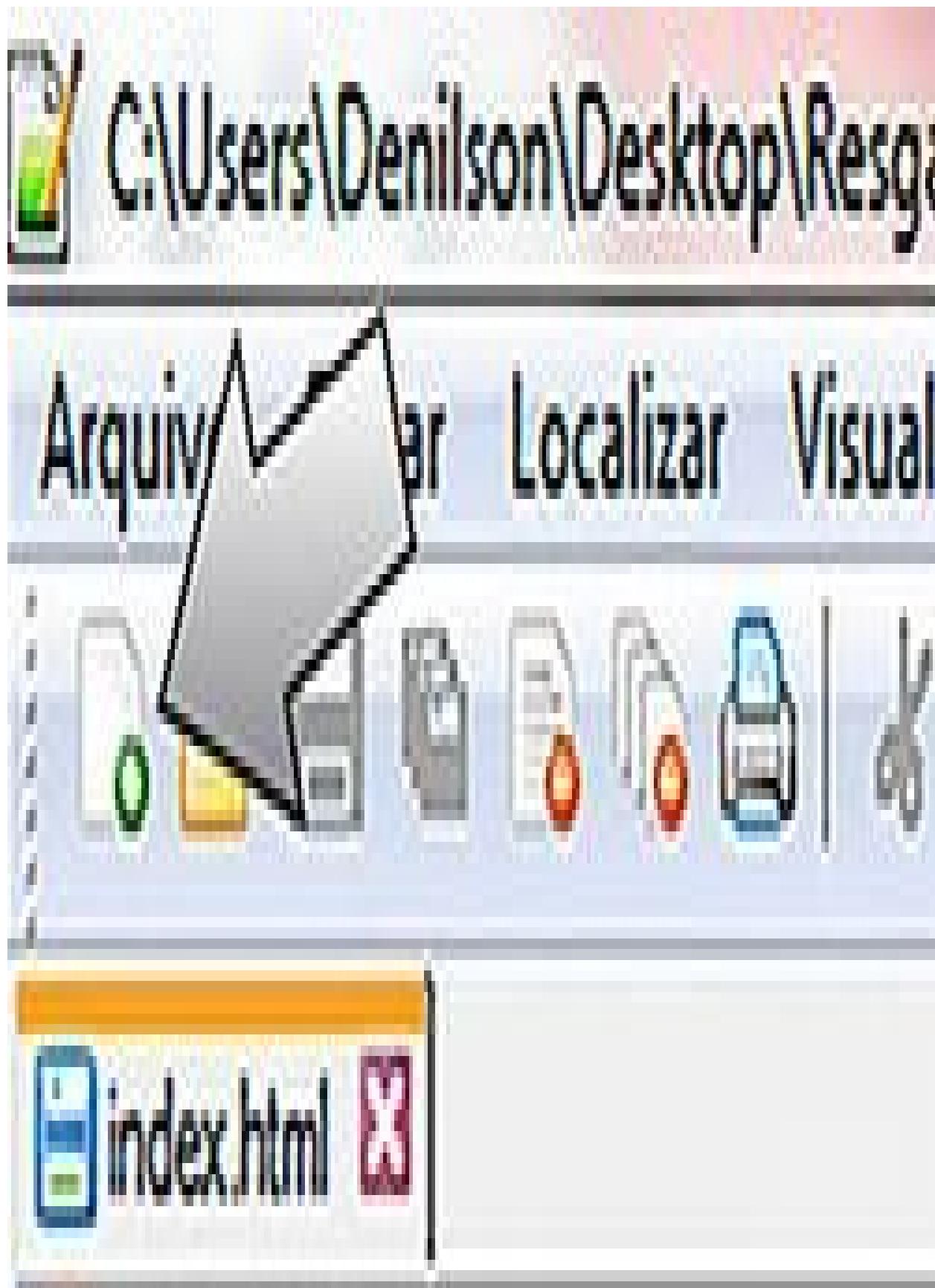
Declaração

## **Figura 19**

- **Seletor:** alvo da regra CSS.
- **Declaração:** parâmetros de estilização.
- **Propriedade:** define qual será a característica do seletor a ser estilizada.
- **Valor:** é a quantidade ou a qualificação da propriedade.

Vamos criar o código CSS que dará formatação às divs criadas no capítulo anterior.

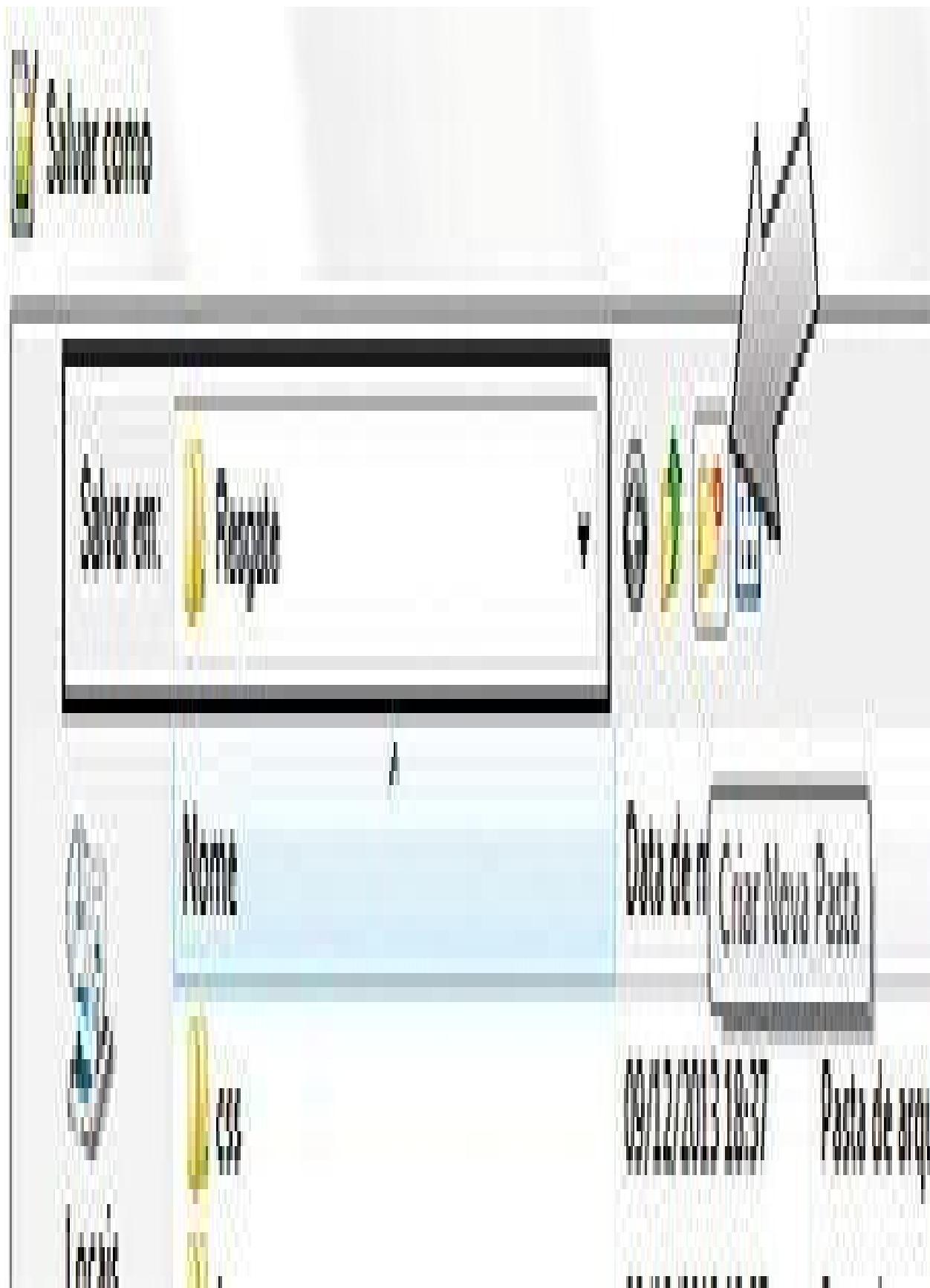
1. Com o Notepad++ aberto, vamos criar um novo arquivo. Dê um clique no botão “Novo”, indicado na Figura 20.



## **Figura 20**

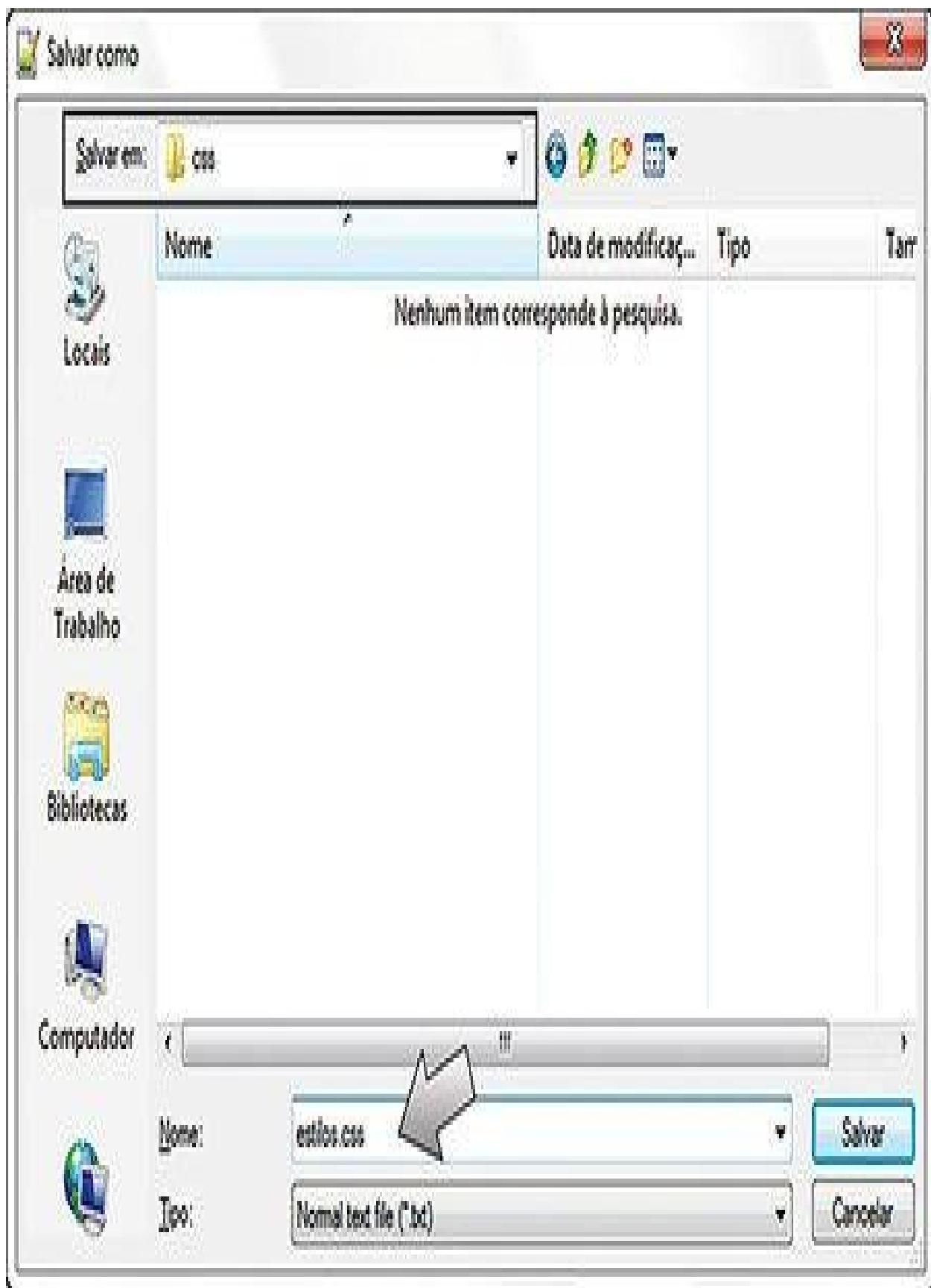
Antes de inserir o código vamos salvar o arquivo. Este arquivo deverá ser salvo dentro de uma pasta com o nome de css dentro da pasta Resgate.

2. Dê um clique no menu Arquivo e selecione a opção “Salvar como”.
3. Abra a pasta Resgate e clique no botão “Criar Nova Pasta”, indicado na Figura 21.



## **Figura 21**

4. Digite o nome css para a nova pasta.
5. Dê um clique duplo na pasta css para abri-la.
6. Salve o arquivo com o nome de estilos.css, como indicado na Figura 22.



## **Figura 22**

Inicialmente no código CSS iremos indicar a fonte que será utilizada em alguns textos do jogo.

7. Digite o código a seguir:

```
@font-face {  
    font-family: Titulo;  
    src: url(..../fontes/ANODETONOONE.TTF);  
}
```

-

Respeite sempre os caracteres maiúsculos e minúsculos na digitação do código.

■ A regra @font-face é utilizada para fontes hospedadas em servidores remotos. Essa funcionalidade resolve o problema do sistema operacional do usuário não ter a fonte utilizada no website.

Neste exemplo estamos utilizando o arquivo ANODETONOONE.TTF, salvo em fontes.

Observe que atribuímos pela propriedade font-family o nome de Titulo para a fonte. Vamos utilizar a fonte Titulo para todos os textos indicados pela tag <h1>.

8. Digite o código a seguir no final do arquivo css:

```
h1 {  
    font-family: Titulo;  
    font-size: 40px;  
    color: #603A03;  
}
```

Veja no código digitado que indicamos para o seletor <h1> a fonte Titulo na propriedade font-family, o tamanho da fonte de 40px pela propriedade font-size e a cor #603A03 pela propriedade color.

O próximo passo é indicar qual imagem será utilizada de fundo. Para isso vamos

atribuir uma imagem para o seletor <body>.

9. Digite o código a seguir no final do arquivo estilos.css:

```
body {  
background-image:url(..../imgs/fundo.jpg);  
}  
}
```

Observe que indicamos na propriedade background-image a imagem fundo.jpg salva na pasta imgs.

A seguir vamos indicar a posição inicial de cada uma das divs criadas no arquivo index.html. Você irá observar que vamos indicar a formatação de algumas divs que ainda não foram criadas no código HTML. Não se preocupe, pois no próximo capítulo iremos criá-las através do código JavaScript.

10. Digite o código a seguir no final do arquivo estilos.css:

```
#container{  
width:950px;  
height:auto;  
position:relative;  
margin-left:auto;
```

```
margin-right: auto;  
}
```

```
#fundoGame {  
width:950px;  
height:630px;  
background-image:url(..../imgs/fundo_game.jpg);  
border-color:#FFF;  
border-style:solid;  
}
```

```
#jogador {  
    width:256px;  
    height:66px;  
    position:absolute;  
    left:8px;  
    top:179px;  
    background-image:url(..../imgs/apache.png);  
}
```

```
#inimigo1 {  
width: 256px;  
height: 66px;  
position: absolute;  
left: 689px;  
top: 253px;  
background-image:url(..../imgs/inimigo1.png);  
}  
  
#inimigo2 {  
width: 165px;  
height: 70px;  
position: absolute;  
left: 775px;  
top: 447px;  
background-image: url(..../imgs/inimigo2.png);  
}  
  
#amigo {  
width: 44px;  
height: 51px;
```

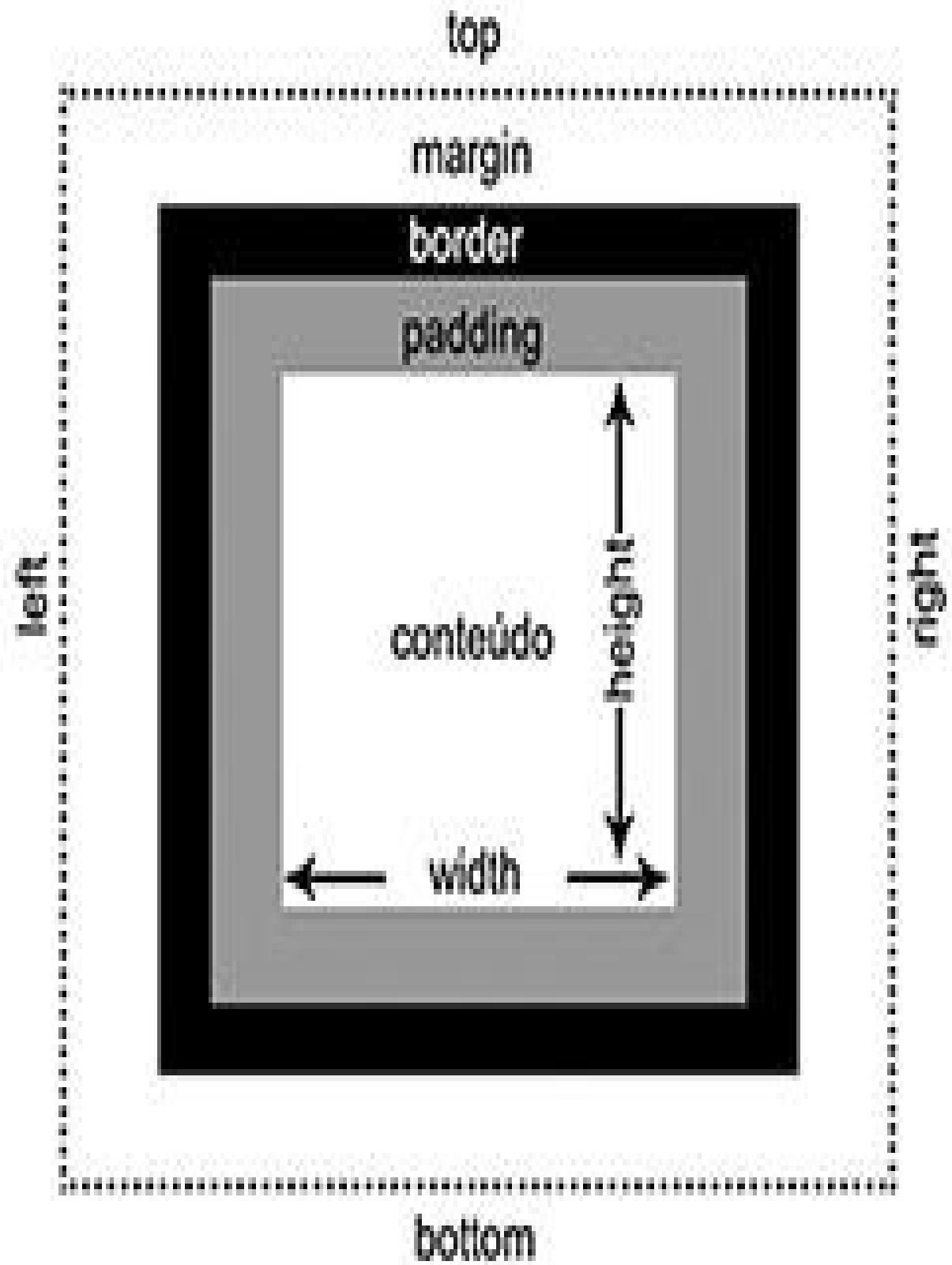
```
position: absolute;  
left: 10px;  
top: 464px;  
background-image:url(..../imgs/amigo.png);  
}
```

```
#inicio {  
width:350px;  
height:200px;  
background-color:#FFF;  
margin-left:auto;  
margin-right:auto;  
margin-top:100px;  
text-align:center;  
padding:10px;  
}
```

A definição da formatação das divs deve seguir o conceito de box model. O box model ou bloco de conteúdo é uma caixa retangular que pode conter um ou mais boxes.

As propriedades CSS que determinam as dimensões dos boxes são: margin,

border, padding, width e height, e as referências para os quatro cantos do box são: top, right, bottom e left. Observe a Figura 23.



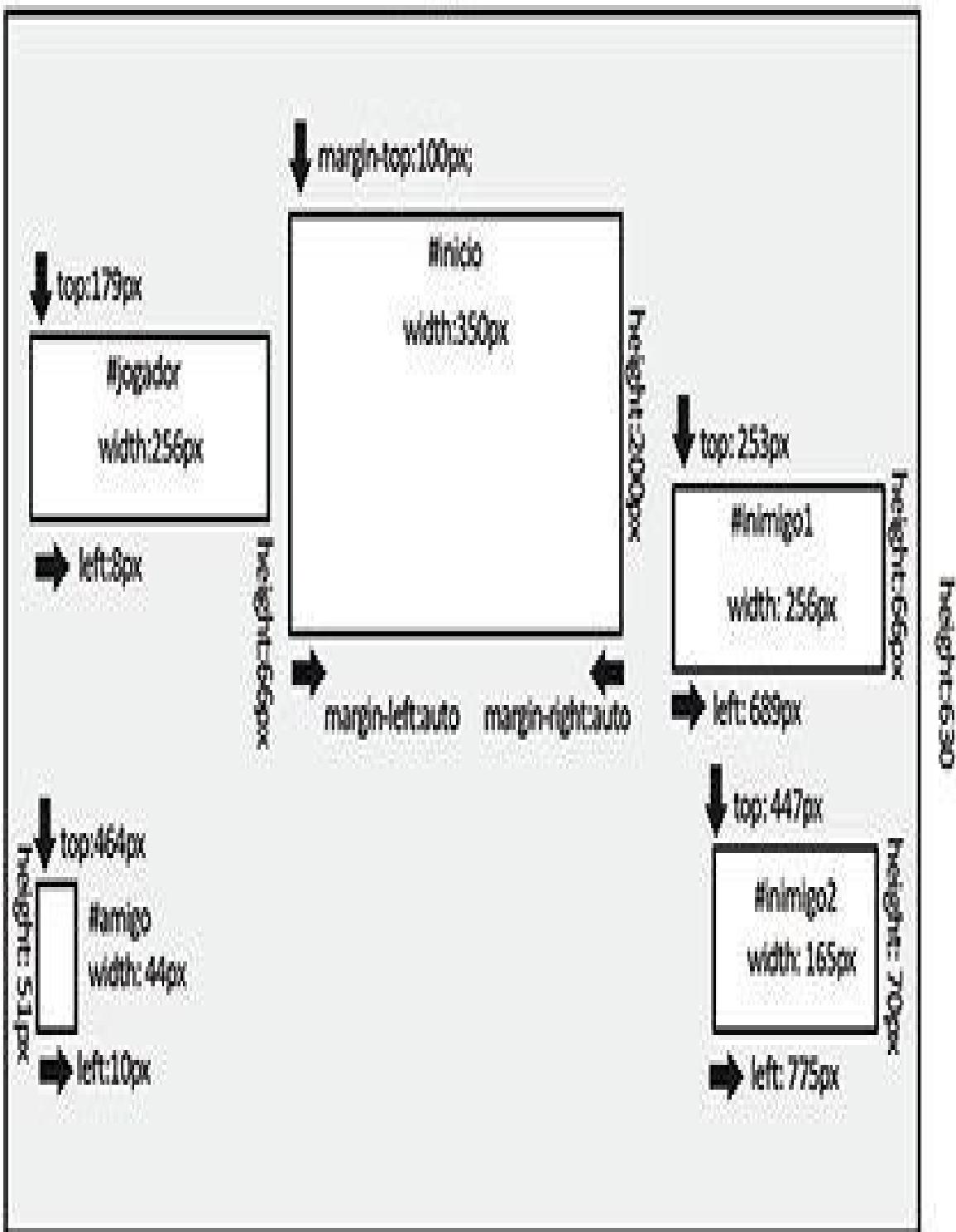
## **Figura 23**

Não há a necessidade de configurar todas as propriedades de um box, mas somente aquelas que você deseja alterar.

Devemos criar a formatação com os nomes dos seletores indicados na tag <div> pela propriedade id: container, jogador, inimigo1 etc. Observe que, pelo código CSS digitado anteriormente, as divs ficarão posicionadas como indicado na Figura 24. É claro que somente as divs que já foram criadas no código HTML serão posicionadas. Mas na imagem a seguir você terá uma ideia de como as divs serão posicionadas inicialmente.

#fundoGame

width:950px



## **Figura 24**

11. Pressione as teclas Ctrl + S para salvar as alterações no arquivo estilos.css.

O próximo passo é vincular o arquivo index.html ao arquivo estilos.css.

12. Abra o arquivo index.html e digite o código a seguir entre as tags <head> e </head>, como indicado na Figura 25.

```
<link href="css/estilos.css" rel="stylesheet" type="text/css">
```

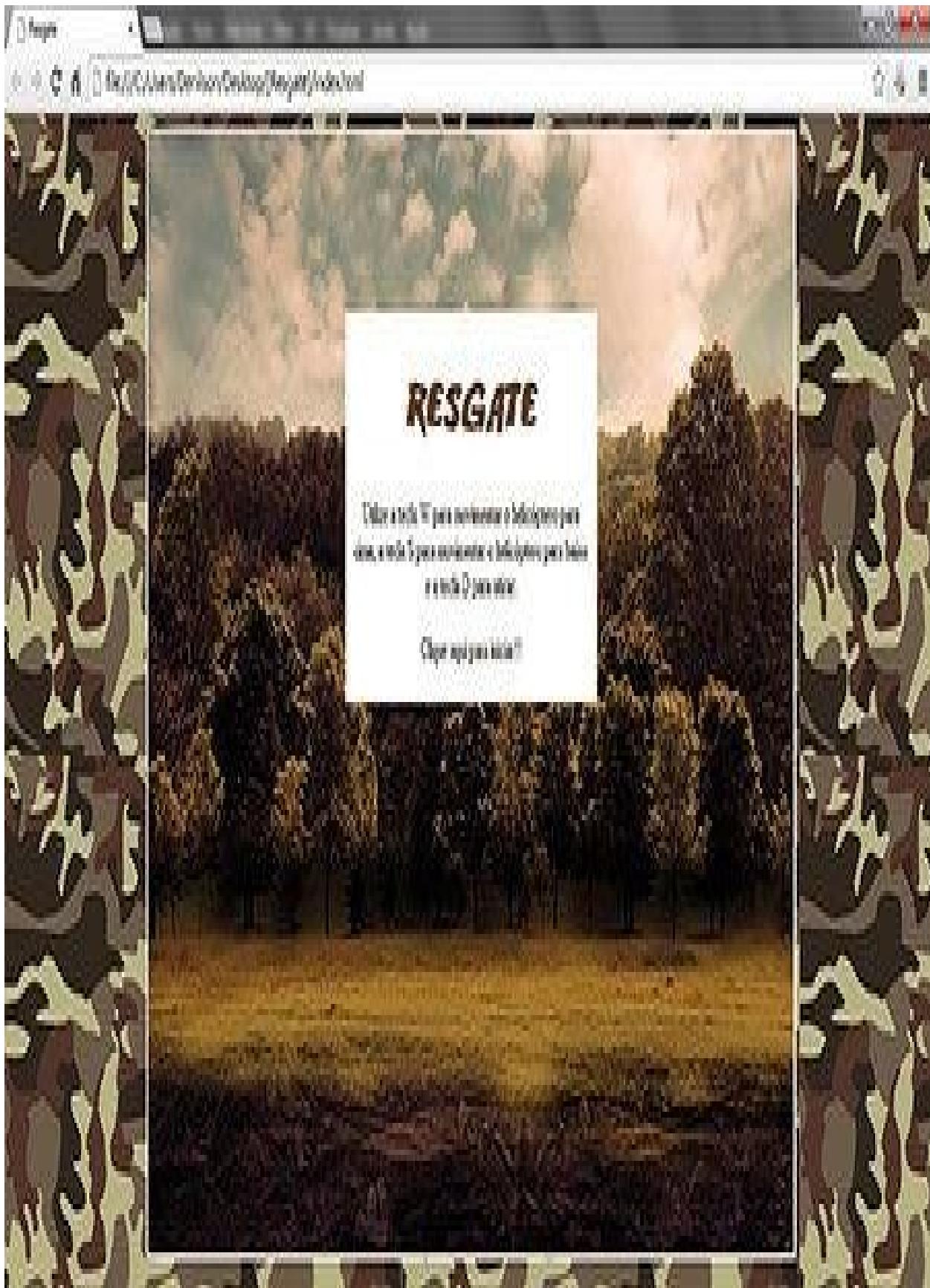
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Sass Test</title>
6     <link href="css/styles.css" rel="stylesheet" type="text/css" />
7   </head>
```

## **Figura 25**

13. Pressione as teclas Ctrl + S para salvar as alterações no arquivo index.html.

14. Abra a pasta Resgate e execute o arquivo index.html no Google Chrome.

Observe que as divs serão exibidas e posicionadas no browser (Figura 26).



**Figura 26**

## Capítulo 4

### Arquivo JavaScript

O JavaScript é uma linguagem de programação com recursos para acessar partes de um documento HTML, incluindo estilos dentro dos elementos CSS.

Juntamente com o framework jQuery, utilizaremos a linguagem JavaScript para indicar o comportamento do jogo, como: movimentação dos objetos, exibição de novos elementos, detecção de teclas pressionadas etc.

O primeiro comportamento que iremos indicar pelo JavaScript é criação das divs jogador, inimigo1, inimigo2 e amigo.

Inicialmente vamos incorporar o framework jQuery ao arquivo index.html e indicar qual será o arquivo JavaScript que conterá o comportamento do jogo.

1. Digite o código a seguir no arquivo index.html, no local indicado pela Figura 27.

```
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
```

```
<script type="text/javascript" src="js/js.js"></script>
```

```
</div> <!-- Fin do div fundante -->  
  
</div> <!-- Fin do div container -->  
  
</body>
```

```
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
```

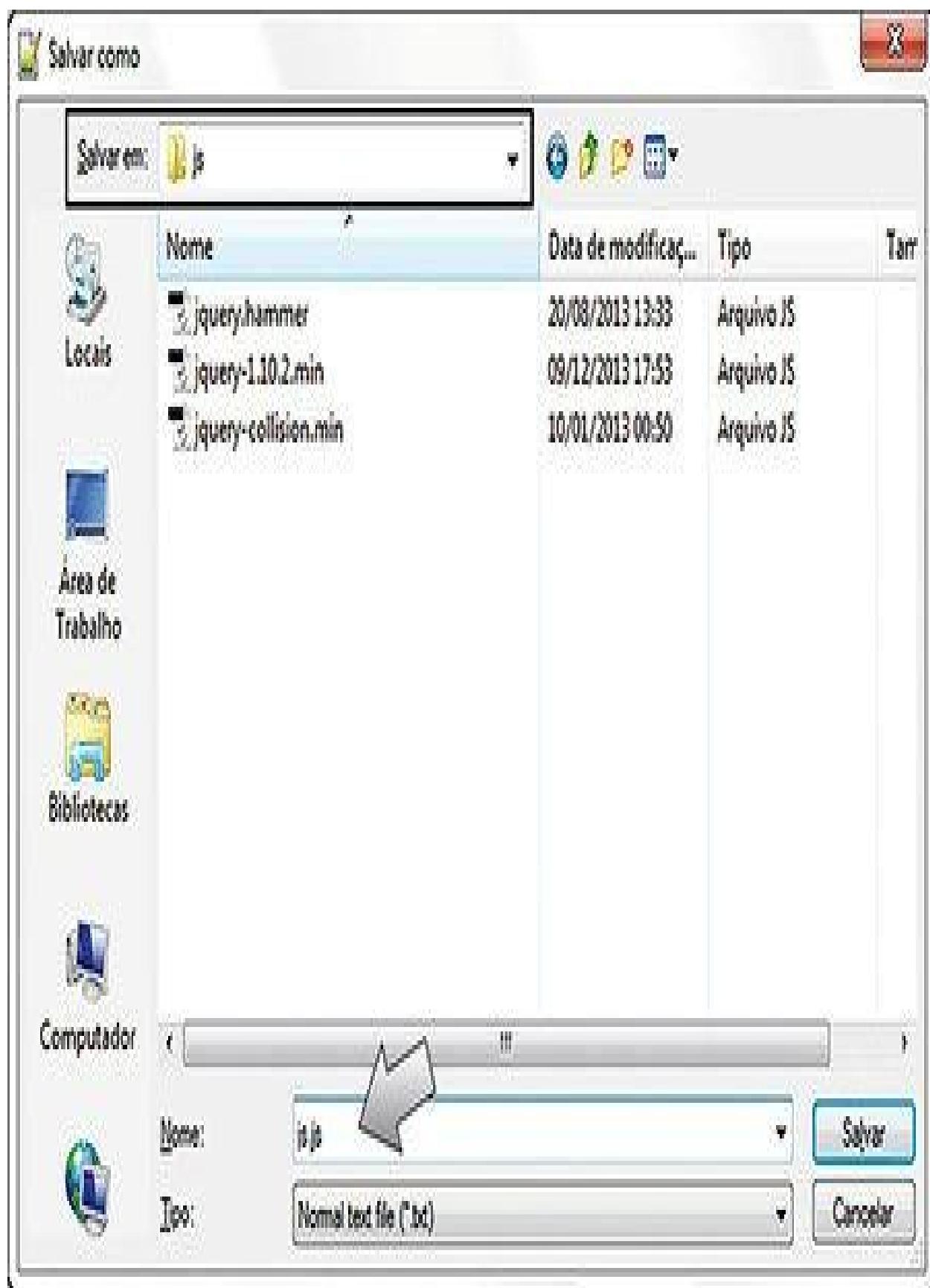
```
<script type="text/javascript" src="js.js"></script>
```

```
</html>
```

## **Figura 27**

O arquivo da biblioteca jQuery jquery-1.10.2.min.js já está salvo na pasta js. Lembre-se de que copiamos a pasta js para a pasta Resgate anteriormente. O próximo passo é criar o arquivo js.js.

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo index.html.
3. Crie um novo arquivo.
4. Salve este arquivo dentro da pasta js com o nome de js.js (Figura 28).



## **Figura 28**

5. Digite o código JavaScript a seguir:

```
function start() { // Início da função start()  
  
    $("#inicio").hide();  
  
    $("#fundoGame").append("<div id='jogador'></div>");  
    $("#fundoGame").append("<div id='inimigo1'></div>");  
    $("#fundoGame").append("<div id='inimigo2'></div>");  
    $("#fundoGame").append("<div id='amigo'></div>");  
  
} // Fim da função start()
```



Utilizando o jQuery, a sintaxe dos comandos JavaScript pode ser simplificada e podemos utilizar várias funções prontas que facilitam a programação do jogo.

Pelo código anterior, criamos uma função com o nome de start. Quando essa função for executada, inicialmente a div inicio será ocultada pela função jQuery hide. Observe o código digitado:

```
$("#inicio").hide();
```

A seguir o código irá criar as divs jogador, inimigo1, inimigo2 e amigo dentro da div fundoGame pela função jQuery append:

```
$("#fundoGame").append("<div id='jogador'></div>");  
$("#fundoGame").append("<div id='inimigo1'></div>");  
$("#fundoGame").append("<div id='inimigo2'></div>");  
$("#fundoGame").append("<div id='amigo'></div>");
```

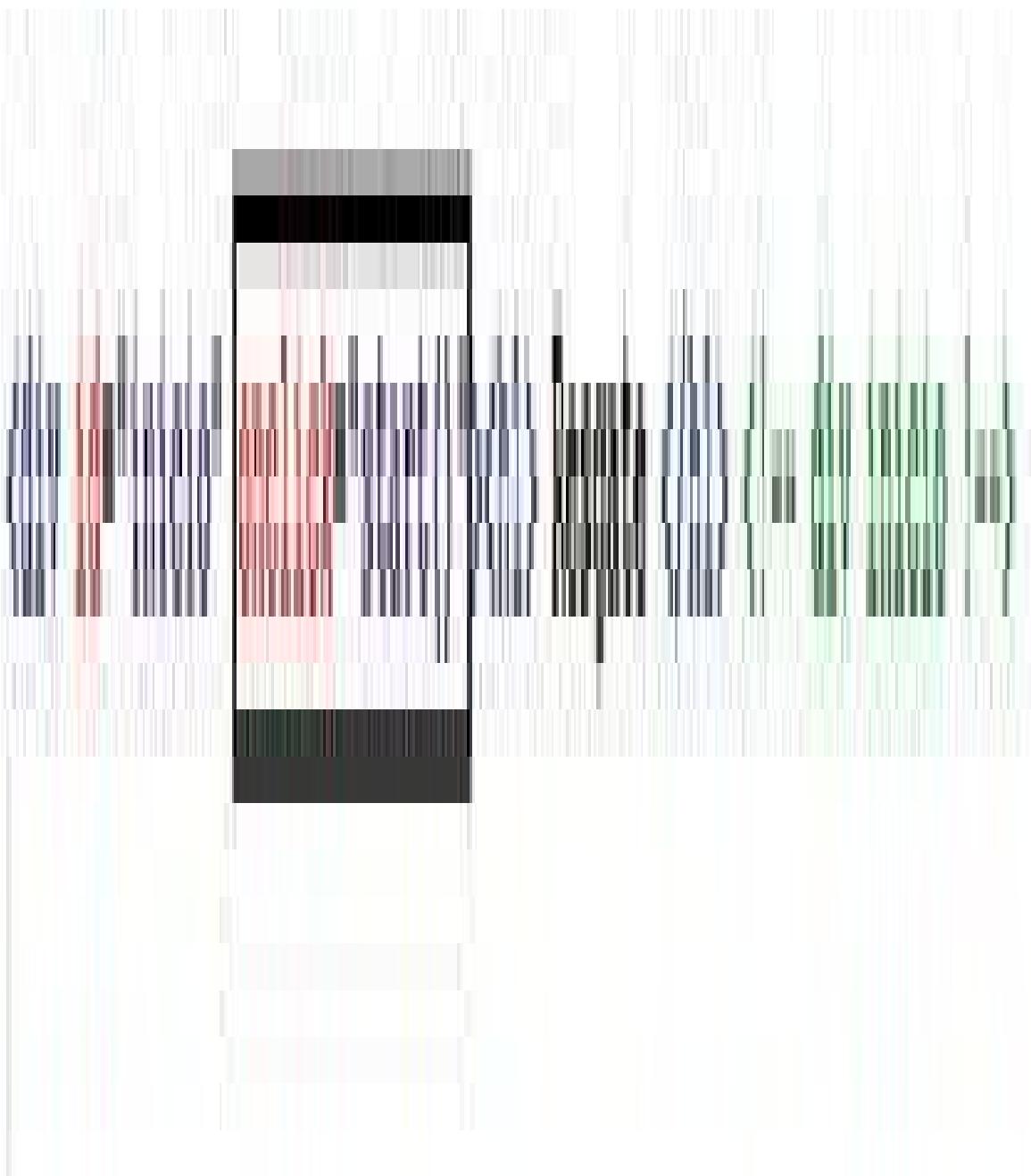
6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é “chamar” a função start quando a div inicio for clicada.

7. No arquivo index.html, digite o código a seguir no local indicado na Figura

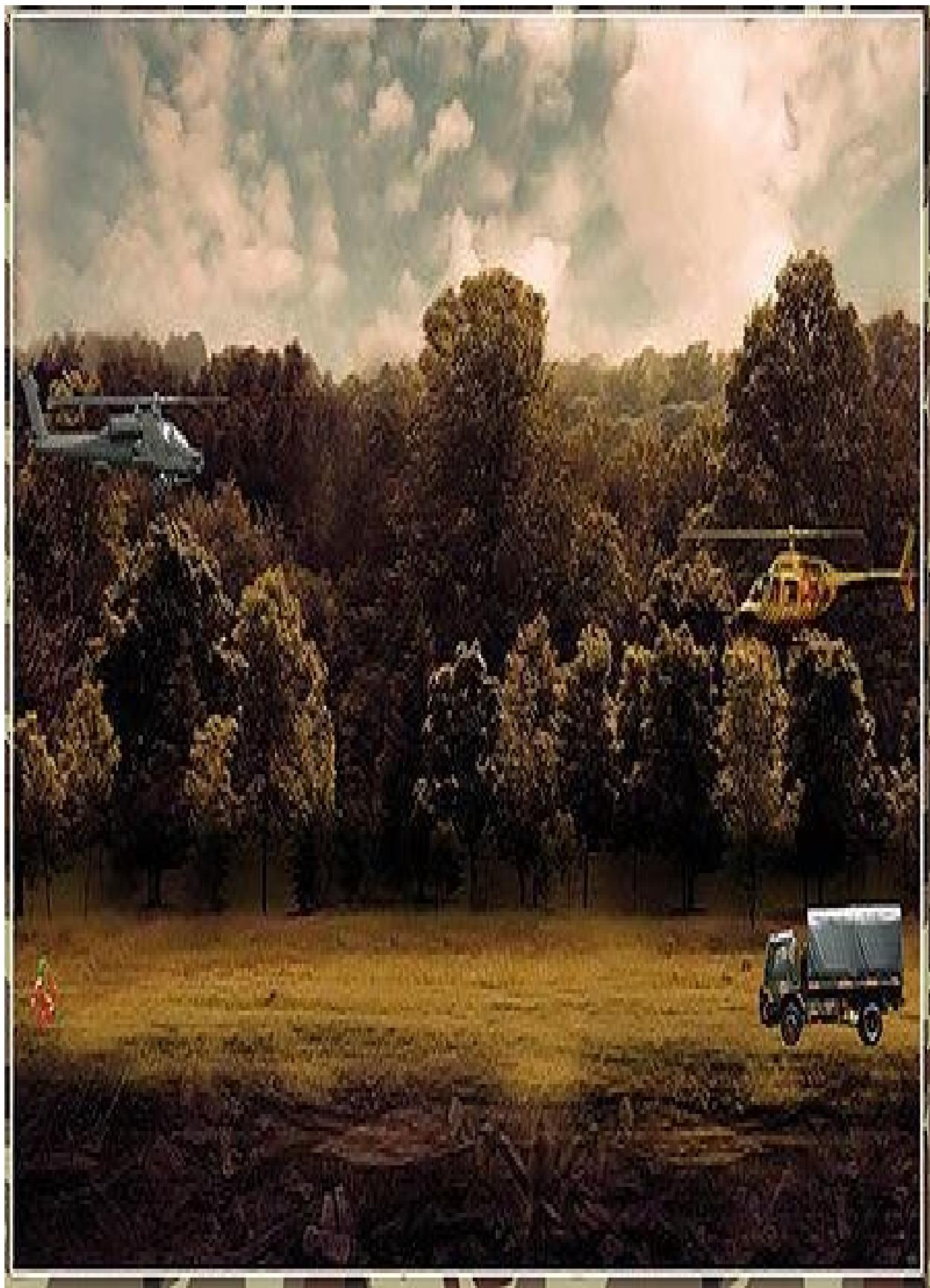
29.

onclick="start()"



### **Figura 29**

8. Pressione o comando Ctrl + S para salvar as alterações no arquivo.
9. Execute o arquivo index.html no Google Chrome. Observe que, ao clicar na div inicio, ela é ocultada e as demais divs são exibidas (Figura 30).



### **Figura 30**

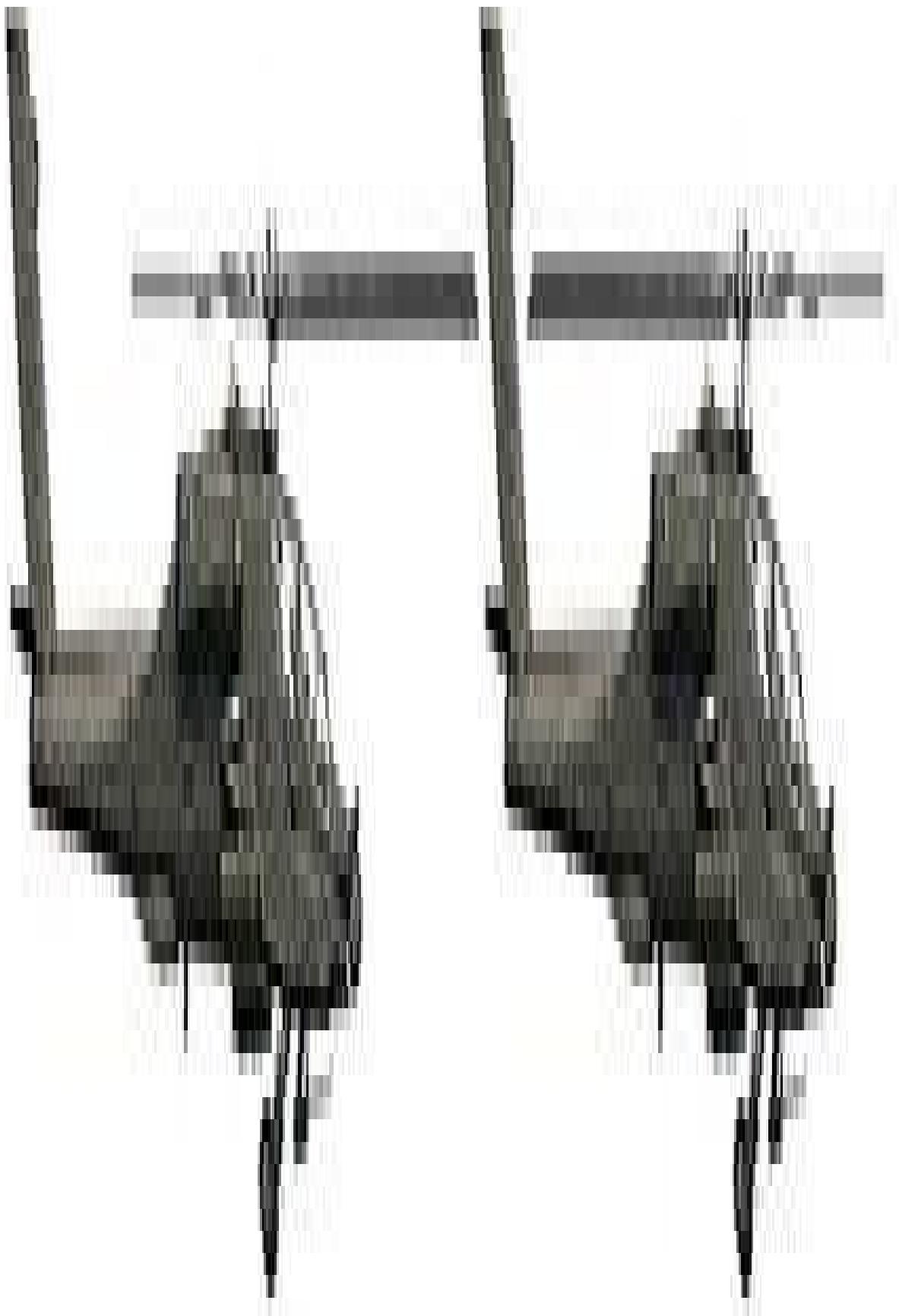
Observe que as divs foram criadas e posicionadas pelo código CSS.

## Capítulo 5

### Criando animações utilizando o CSS3

Para que o jogo fique mais dinâmico e atraente criaremos algumas animações. Iremos utilizar uma imagem conhecida como sprite.

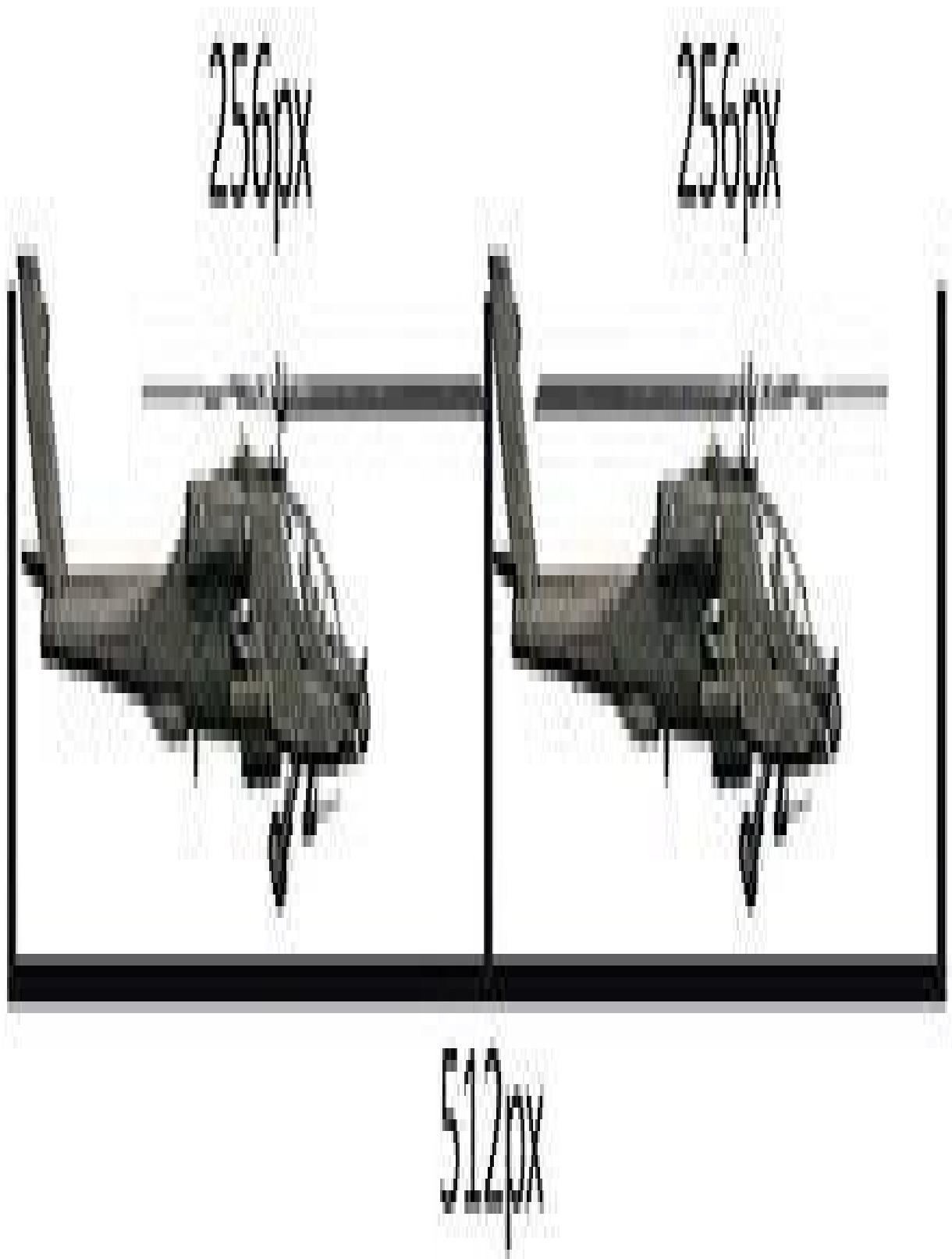
*Sprites são desenhos, em sua maioria em duas dimensões (2D), agrupados em uma mesma imagem. Essas imagens ao serem exibidas individualmente geram uma animação. Na Figura 31 temos a imagem em forma de sprite que iremos utilizar na primeira animação:*



## **Figura 31**

Observe que em uma mesma imagem temos o mesmo helicóptero com as hélices em uma posição diferente. Cada parte dessa imagem será exibida em uma sequência, criando a animação.

Essa imagem possui a largura total de 512px. Exibiremos inicialmente a primeira parte da imagem com 256px, criando o primeiro quadro-chave (keyframe) da animação e posteriormente exibiremos a segunda parte da imagem com o mesmo tamanho de 256px, criando o segundo quadro-chave da animação. Veja a Figura 32.



## **Figura 32**

Vamos fazer o código CSS3 que irá criar essa animação.

### 1. Abra o arquivo estilos.css.

Para criar a animação utilizaremos uma classe CSS onde teremos a propriedade `animation`. Nela iremos indicar o nome da diretiva que terá a marcação dos quadros-chave da animação. Nesse primeiro exemplo, chamaremos a diretiva de `play`.

Daremos o nome de `anima1` para a classe CSS.

### 2. Digite o código a seguir no final do arquivo:

```
.anima1 {  
    width:256px;  
    height:66px;  
    background-image:url(..imgs/helicoptero.png);  
    animation:play .5s steps(2) infinite;  
    -webkit-animation: play .5s steps(2) infinite;  
    -moz-animation: play .5s steps(2) infinite;
```

```
-ms-animation: play .5s steps(2) infinite;  
-o-animation: play .5s steps(2) infinite;  
}
```

Observe que, além de utilizar a propriedade CSS3 animation, utilizamos os prefixos -webkit, -moz, -ms e -o antes da definição da propriedade animation. Não existe ainda compatibilidade total de recursos CSS3 em todos os browsers. Para o perfeito funcionamento da animação devemos utilizar os prefixos proprietários: -moz, para Firefox; -webkit, para o Chrome; e -o- e -ms para Opera e Internet Explorer. Para uma consulta sobre compatibilidade de recursos CSS3 em várias versões de browsers acesse o link a seguir:

[http://www.w3schools.com/cssref/css3\\_browsersupport.asp](http://www.w3schools.com/cssref/css3_browsersupport.asp)

Veja que na classe anima1 indicamos as seguintes propriedades:

Propriedade	Descrição
width:256px	Largura da div. Observe que foi utilizada a largura de cada
height:66px	Altura de cada um dos quadros- chave.
background-image	Imagen que será utilizada na animação.
animation:play	Indicamos que o gerenciador de quadros-chave terá o nom
.5s	Tempo em que cada um dos quadros-chave serão exibidos.
steps(2)	Número total de quadros-chave.

infinite

Indicamos que a animação deverá ser exibida em um loop

■

O próximo passo é criar a diretiva @keyframes com o nome de play.

3. Digite o código a seguir:

```
@keyframes play {  
from {background-position:0px;}  
to {background-position:-512px;}  
}
```

```
@-webkit-keyframes play {  
from { background-position:0px; }  
to { background-position: -512px; }  
}
```

```
@-moz-keyframes play {  
from { background-position:0px; }  
to { background-position: -512px; }  
}
```

```
@-ms-keyframes play {
```

```
from { background-position:0px; }

to { background-position: -512px; }

}
```

```
@-o-keyframes play {

from { background-position:0px; }

to { background-position: -512px; }

}
```

Na propriedade from, indicamos que no primeiro quadro-chave da animação o fundo da div, especificado pela propriedade CSS background-position, deverá estar posicionada em 0px. No segundo quadro, pela propriedade to, indicamos que a mesma imagem deverá estar posicionada em -512px.

Observe também que, ao utilizarmos todos os prefixos proprietários para a diretiva @keyframes, evitamos qualquer tipo de incompatibilidade entre os browsers.

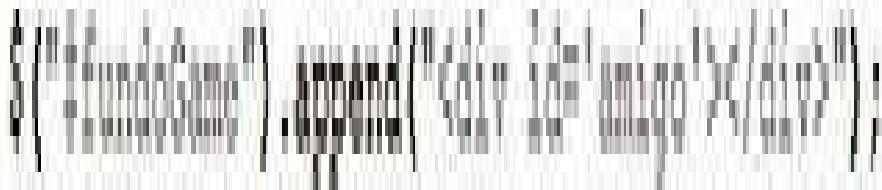
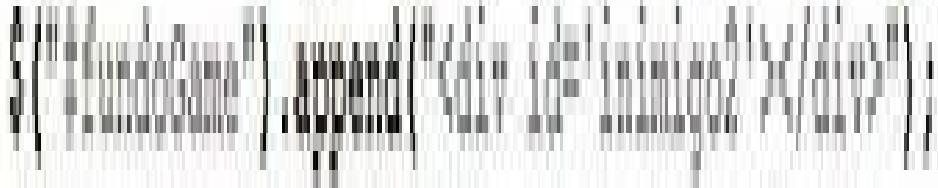
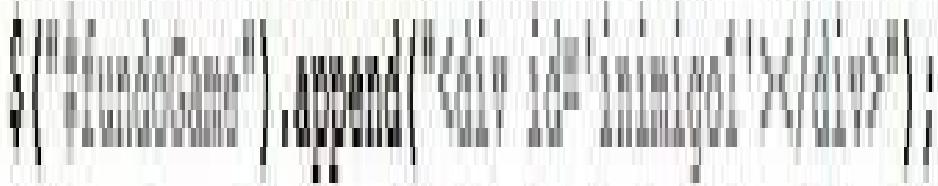
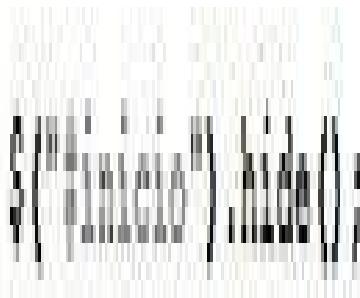
4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo estilos.css.

O próximo passo é vincular a div jogador à classe anima1.

5. Abra o arquivo js.js.

6. Digite o código a seguir no local indicado na Figura 33.

class='anima1'



### **Figura 33**

7. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
8. Execute o arquivo index.html no Google Chrome. Observe que quando o jogo for iniciado a animação na div jogador será executada.

O próximo passo é criar a animação para a div inimigo1. Vamos criar uma classe CSS com o nome de anima2.

Abra o arquivo estilos.css.

9. Digite o código a seguir no final do arquivo:

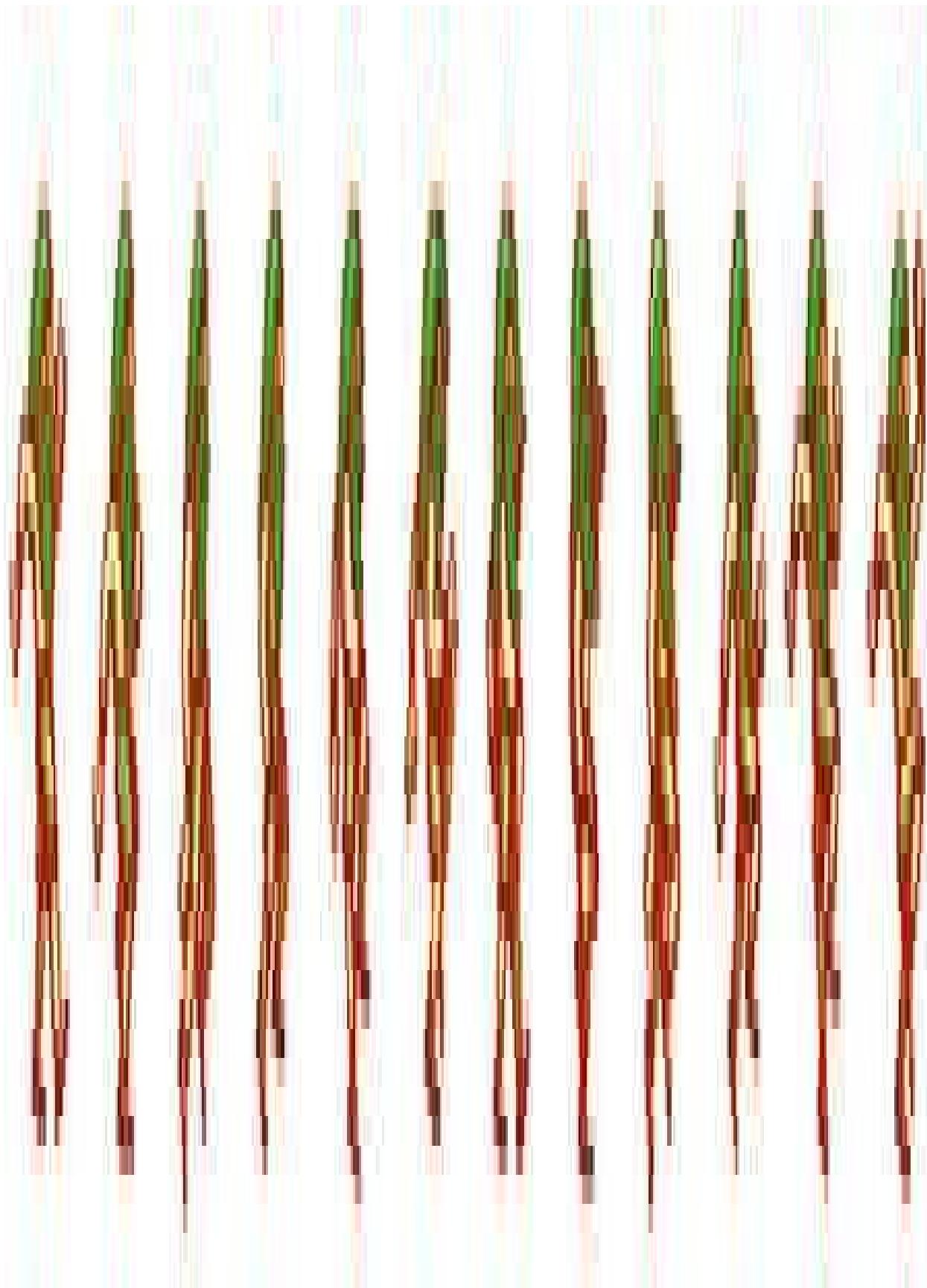
```
.anima2 {  
    width: 256px;  
    height: 66px;  
    background-image:url(..../imgs/inimigo1.png);  
    animation: play .5s steps(2) infinite;  
    -webkit-animation: play .5s steps(2) infinite;  
    -moz-animation: play .5s steps(2) infinite;  
    -ms-animation: play .5s steps(2) infinite;
```

```
-o-animation: play .5s steps(2) infinite;
```

```
}
```

Veja que nesta classe apenas alteramos o nome do arquivo de fundo da div (que será utilizado na animação) para inimigo1.jpg e mantemos o mesmo gerenciador de quadros-chave: play.

O próximo passo é criar a animação que será exibida na div amigo. Para essa animação vamos usar o arquivo amigo.jpg (Figura 34).



## Figura 34

Iremos criar essa animação com doze quadros-chave de tamanho 44px de largura cada um.

10. Digite o código a seguir no final do arquivo estilos.css:

```
.anima3 {  
    width:44px;  
    height:51px;  
    background-image:url(..../imgs/amigo.png);  
    animation:play2 .9s steps(12) infinite;  
    -webkit-animation: play2 .9s steps(12) infinite;  
    -moz-animation: play2 .9s steps(12) infinite;  
    -ms-animation: play2 .9s steps(12) infinite;  
    -o-animation: play2 .9s steps(12) infinite;  
}
```

Observe que na classe anima3 indicamos as seguintes propriedades:

-

Propriedade	Descrição
width:44px	Largura da div. Observe que utilizamos a largura que cada
height:51px	Altura de cada um dos quadros-chave.
background-image	Imagen que será utilizada na animação.
animation:play2	Indicamos que o gerenciador de quadros-chave terá o nom
.9s	Tempo que um dos quadros-chave será exibido.
steps(12)	Número total de quadros-chave.
infinite	Indicamos que a animação deverá ser exibida em um loop

■

O próximo passo é criar o gerenciador de keyframes play2.

11. Digite o código a seguir no final do arquivo.

```
@keyframes play2 {  
from {background-position:0px;}  
to {background-position:-528px;}  
}
```

```
@-webkit-keyframes play2 {  
from { background-position: 0px; }  
to { background-position: -528px; }  
}
```

```
@-moz-keyframes play2 {  
from { background-position: 0px; }  
to { background-position: -528px; }  
}
```

```
@-ms-keyframes play2 {
```

```
from { background-position: 0px; }

to { background-position: -528px; }

}
```

```
@-o-keyframes play2 {

from { background-position: 0px; }

to { background-position: -528px; }

}
```

Veja que indicamos a posição inicial de 0px da imagem na animação pela propriedade from e a posição final de -528px da imagem na animação pela propriedade to.

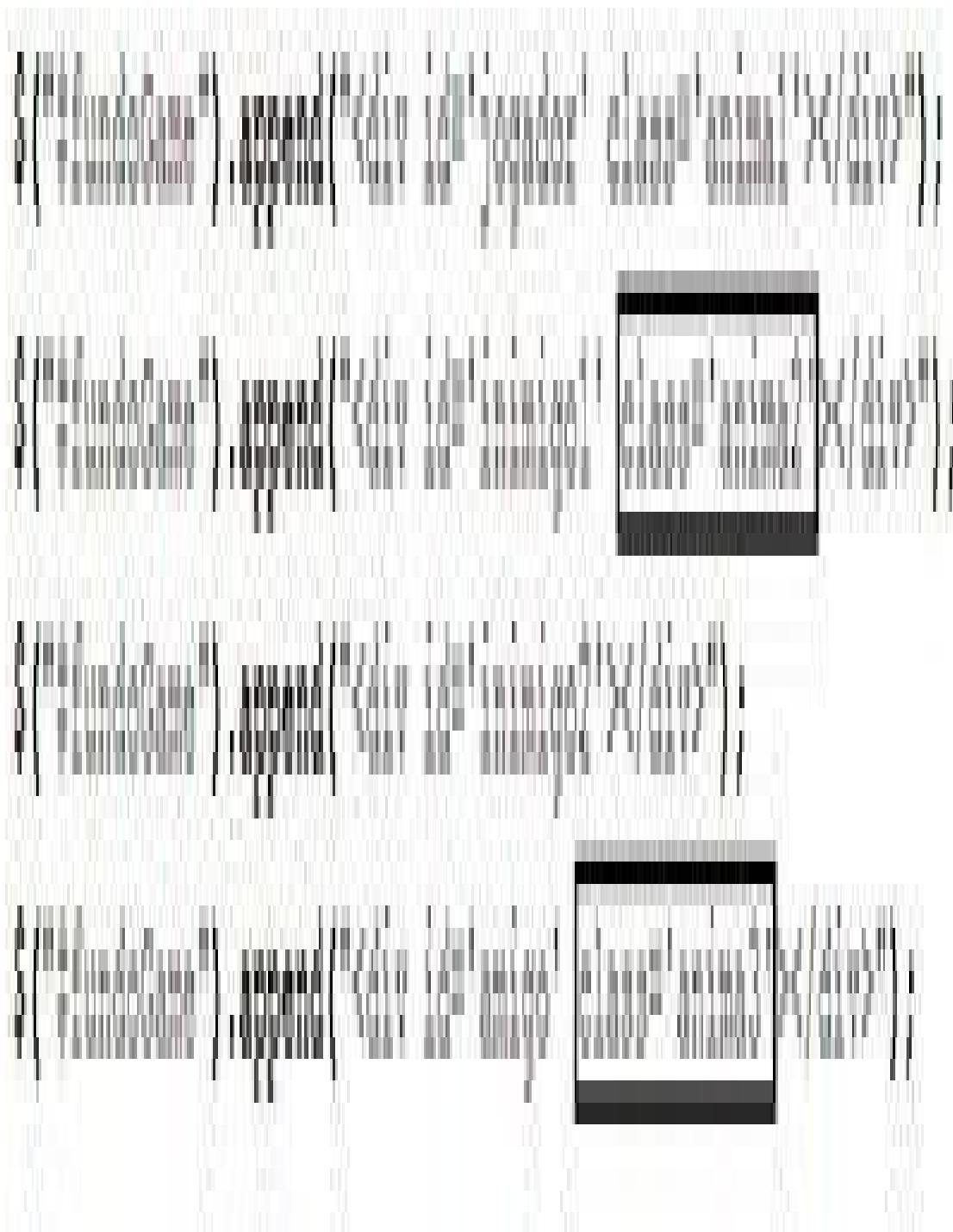
12. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é vincular a classe anima2 à div inimigo1 e a classe anima3 à div amigo.

13. No arquivo js.js, digite o código a seguir no local indicado na Figura 35.

```
class='anima2'
```

class='anima3'



### **Figura 35**

14. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
15. Execute o arquivo index.html no Google Chrome. Observe que quando o jogo for iniciado a animação na div inimigo1 e amigo será executada.

## Capítulo 6

### ***Game Loop***

O game loop é uma função executada constantemente (em um loop) que irá “chamar” as principais funções do jogo, como: captura de teclas pressionadas, movimentar o fundo do jogo, movimentar os inimigos, detectar colisões etc.

Esta será a principal função do jogo; parando essa função, as demais não serão mais executadas e o jogo será interrompido.

Vamos ao código que criará o game loop.

1. Abra o arquivo js.js.
2. Digite o código a seguir dentro da função start no local indicado na Figura 36.

```
//Principais variáveis do jogo
```

```
var jogo = {}
```

```
//Game Loop
```

```
jogo.timer = setInterval(loop,30);
```

```
function loop() {
```

```
movefundo();
```

```
} // Fim da função loop()
```

```
$( "#fundoGame" ).append("<div id='jogador' class='animal'></div>");  
$( "#fundoGame" ).append("<div id='inimigo1' class='animal'></div>");  
$( "#fundoGame" ).append("<div id='inimigo2'></div>");  
$( "#fundoGame" ).append("<div id='amigo' class='animal'></div>");
```

```
//Principais variáveis do jogo
```

```
var jogo = {}
```

```
//Game Loop
```

```
jogo.timer = setInterval(loop,30);
```

```
function loop() {
```

```
movefundo();
```

```
} // Fim da função loop()
```

```
} // Fim da função start
```

**Figura 36**



Veja que inicialmente criamos uma variável com o nome de jogo.

```
var jogo = {}
```

Na variável jogo criamos uma propriedade com o nome de timer que irá executar uma função com o nome de loop a cada trinta milissegundos.

```
jogo.timer = setInterval(loop,30);
```

O comando setInterval é utilizado para indicar a unidade de tempo em que um determinado evento irá acontecer. Nesse exemplo, a função loop será executada a cada trinta milissegundos.

Observe que na função loop estamos “chamando” uma função com o nome de movefundo. Vamos criar a função movefundo.

3. Digite o código a seguir no local indicado pela Figura 37.

```
//Função que movimenta o fundo do jogo
```

```
function movefundo() {
```

```
esquerda = parseInt($("#fundoGame").css("background-position"));
```

```
$("#fundoGame").css("background-position",esquerda-1);
```

```
} // fim da função movefundo()
```

```
//Game Loop

jogo.timer = setInterval(loop,30);

function loop() {

    movefundo();

} // Fim da função loop()
```

```
//Função que movimenta o fundo do jogo

function movefundo() {

    esquerda = parseInt($("#fundoGame").css("background-position"));
    $("#fundoGame").css("background-position",esquerda-1);

} // fim da função movefundo()
```

```
} // Fim da função start
```

## Figura 37

Na função movefundo que é executada constantemente (a cada trinta milissegundos), inicialmente armazenamos na variável esquerda a posição atual da imagem de fundo div fundoGame pela propriedade CSS background-position da div fundoGame.

```
esquerda = parseInt($("#fundoGame").css("background-position"));
```

Com o valor atual da posição da imagem de fundo armazenada na variável esquerda, na sequência alteramos a posição atual da imagem na div com o valor da variável esquerda menos uma unidade.

```
$("#fundoGame").css("background-position",esquerda-1);
```

Como a função é executada constantemente chamada pelo game loop do jogo, a imagem de fundo da div fundoGame ficará em constante movimentação. Vamos visualizar o resultado.

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html no Google Chrome. Observe que a animação da imagem de fundo foi criada.

# **Não está funcionando, e agora?**

Caso o seu jogo não esteja funcionando corretamente, algumas tarefas podem ser realizadas para resolver o problema.

## **a) Limpe o cache do seu navegador**

Os arquivos do jogo podem estar no cache do navegador e as atualizações nos arquivos podem não estar sendo executadas. Para limpar o cache do navegador Google Chrome utilize os passos especificados no link a seguir:

<https://support.google.com/chrome/answer/95582>

## **b) Tenha certeza de que você salvou os arquivos**

Certifique-se de que você salvou as atualizações em todos os arquivos do jogo. Lembre-se de que você possui três arquivos principais: index.html, estilos.css e js.js.

## **c) Erros de sintaxe**

Esse são erros na digitação do código. Você pode consultar os erros de sintaxe no Google Chrome pela ferramenta Developer Tools.

### **1. Execute o arquivo index.html pelo Google Chrome.**

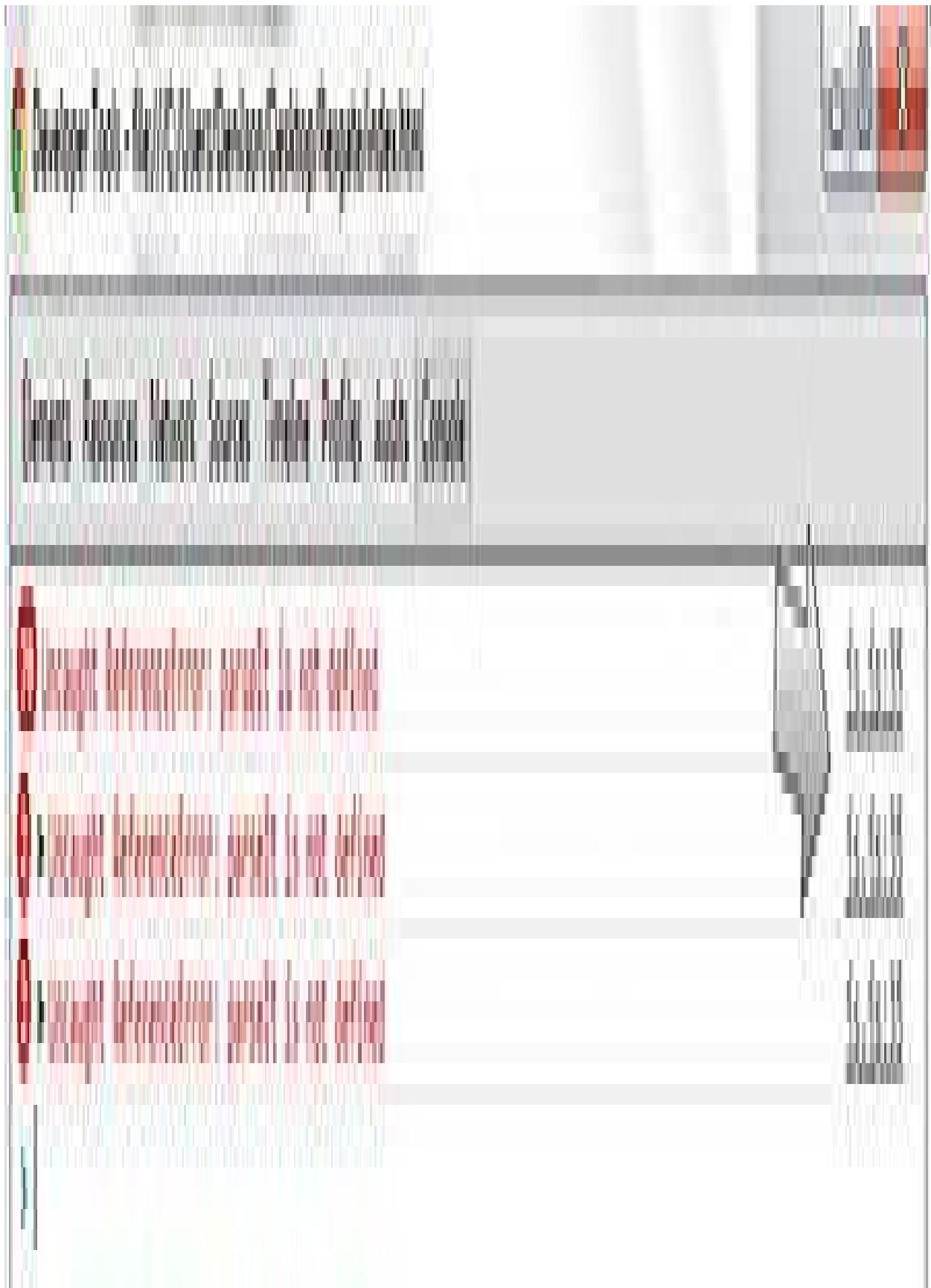
**2. Utilize as teclas Ctrl + Alt + I para abrir a janela Developer Tools.**

**3. Clique sobre a opção “Console” (Figura 38).**



### **Figura 38**

Caso haja algum erro de digitação no código, a ferramenta indicará a linha onde está o erro no arquivo JavaScript. Veja o exemplo na Figura 39.



**Figura 39**

## Capítulo 7

### Detectando teclas pressionadas

A próxima etapa é criar uma função que movimentará o jogador. Utilizaremos as teclas W para movimentar o helicóptero do jogador para cima, a tecla S para movimentar para baixo e a tecla D para realizar os disparos.

1. No arquivo js.js digite o código a seguir no local indicado pela Figura 40.

```
var TECLA = {
```

```
W: 87,
```

```
S: 83,
```

```
D: 68
```

```
}
```

```
jogo.pressionou = [];
```

```
//Principais variáveis do jogo
```

```
var jogo = {
```

```
  var TECLA = {
```

```
    W: 87,
```

```
    S: 83,
```

```
    D: 68
```

```
}
```

```
  jogo.pressionou = [];
```

```
//Game Loop
```

## Figura 40

No código anterior criamos uma variável com o nome de TECLA com os valores decimais de cada uma das teclas que iremos identificar.

Pelo que você deve ter percebido, o valor decimal da tecla W é 87, o da tecla S é 83 e o da tecla D é 68. Caso queira consultar o valor decimal de outras teclas, utilize o seguinte link para consulta:

<http://www.denilsonbonatti.com.br/livros keycode.pdf>

Criamos também uma propriedade com o nome de pressionou para a variável jogo.

Agora iremos criar o código que identificará se o jogador pressionou alguma tecla. Caso o jogador tenha pressionado qualquer tecla, a propriedade pressionou da variável jogo receberá o valor true, caso contrário ela receberá o valor false.

2. Digite o código a seguir no local indicado pela Figura 41.

```
//Verifica se o usuário pressionou alguma tecla
```

```
jogo.pressionou = [];
```

```
$(document).keydown(function(e){
```

```
    jogo.pressionou[e.which] = true;
```

```
});
```

```
$(document).keyup(function(e){
```

```
    jogo.pressionou[e.which] = false;
```

```
});
```

```
var TECLA = {  
    W: 87,  
    S: 83,  
    D: 68  
}
```

```
jogo.pressionou = [];
```

```
//Verifica se o usuário pressionou alguma tecla  
jogo.pressionou = [];  
  
$(document).keydown(function(e) {  
    jogo.pressionou[e.which] = true;  
});
```

## **Figura 41**

O próximo passo é chamar uma função com o nome de movejogador dentro do game loop do jogo.

3. Digite o código a seguir no local indicado na Figura 42.

```
movejogador();
```

```
function loop() {  
    movefundo();  
    movejogador();  
}  
// Fim da função loop()
```

## Figura 42

4. Agora vamos criar a função movejogador. Digite o código a seguir no local indicado pela Figura 43.

```
function movejogador() {  
  if (jogo.pressionou[TECLA.W]) {  
    var topo = parseInt($("#jogador").css("top"));  
    $("#jogador").css("top",topo-10);  
  }  
  
  if (jogo.pressionou[TECLA.S]) {  
    var topo = parseInt($("#jogador").css("top"));  
    $("#jogador").css("top",topo+10);  
  }  
  
  if (jogo.pressionou[TECLA.D]) {  
    //Chama função Disparo  
  }  
}
```

}

} // fim da função movejogador()

```
    } // fim da função movefundo()
```

```
function movejogador() {  
  
    if (jogo.pressionou[TECLA.W]) {  
        var topo = parseInt($("#jogador").css("top"));  
        $("#jogador").css("top",topo-10);  
  
    }  
  
    if (jogo.pressionou[TECLA.S]) {  
  
        var topo = parseInt($("#jogador").css("top"));  
        $("#jogador").css("top",topo+10);  
    }  
  
    if (jogo.pressionou[TECLA.D]) {  
  
        //Chama função Disparo  
    }  
  
} // fim da função movejogador()
```

```
} // Fim da função start
```

### Figura 43

Observe que, na função movejogador, caso o jogador pressione a tecla W, é armazenada na variável topo a posição atual da propriedade top da div jogador. Na linha a seguir essa posição é subtraída em -10 unidades, fazendo com que a div seja movimentada para cima.

```
if (jogo.pressionou[TECLA.W]) {  
  
    var topo = parseInt($("#jogador").css("top"));  
  
    $("#jogador").css("top",topo-10);  
  
}
```

Caso o jogador pressione a tecla S, é armazenada na variável topo a posição atual da propriedade top da div jogador. Na linha a seguir essa posição é somada em 10 unidades, fazendo com que a div seja movimentada para baixo.

```
if (jogo.pressionou[TECLA.S]) {  
  
    var topo = parseInt($("#jogador").css("top"));  
  
    $("#jogador").css("top",topo+10);  
  
}
```

```
$("#jogador").css("top",topo+10);  
}
```

Caso o jogador pressione a tecla D futuramente, será chamada a função que realiza o disparo.

```
if (jogo.pressionou[TECLA.D]) {
```

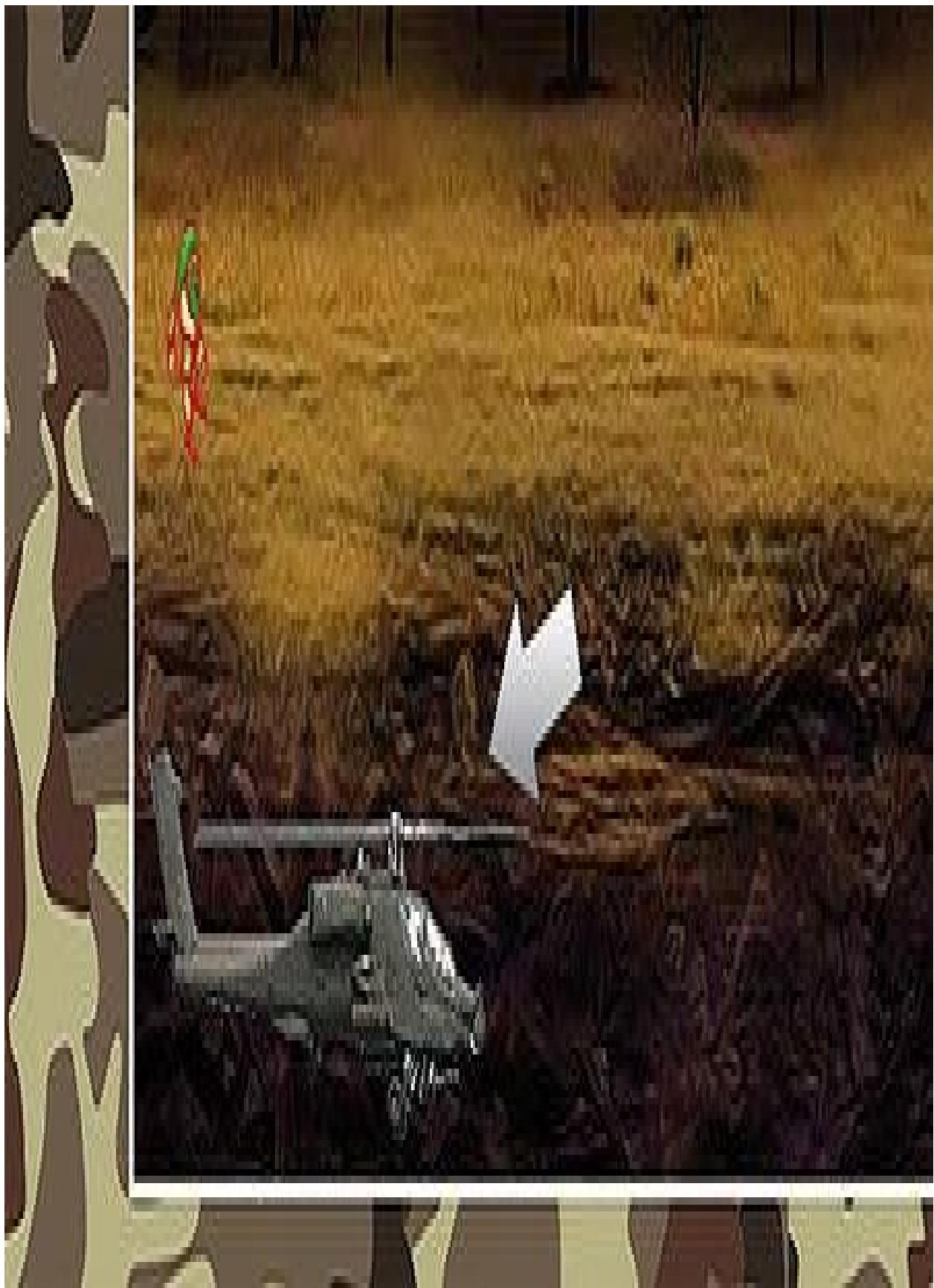
```
    //Chama função Disparo
```

```
}
```

5. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

6. Execute o arquivo index.html pelo Google Chrome.

Observe que, ao pressionar as teclas W e S, a div jogador é movimentada, mas ainda não há um limite de movimentação (Figura 44).



**Figura 44**

## Capítulo 8

# Limitando a movimentação do jogador e criando uma animação pelo JavaScript

Neste capítulo iremos limitar a movimentação da div jogador dentro da div fundogame e daremos início à animação do primeiro inimigo do jogo.

1. Abra o arquivo js.js.

Inicialmente vamos limitar a movimentação do jogador quando a tecla W é pressionada.

2. Digite o código a seguir dentro da função movejogador no local indicado pela Figura 45.

```
if (topo<=0) {  
  
    $("#jogador").css("top",topo+10);  
  
}
```



```
if (jogo.pressionou[TECLA.W]) {  
    var topo = parseInt($("#jogador").css("top"));  
    $("#jogador").css("top", topo-10);  
}  
}  
}  
}
```

```
if (topo<=0) {  
    topo = 0;  
    $("#jogador").css("top", topo+10);  
}  
}
```

## **Figura 45**

No código anterior, caso o valor da variável topo seja menor ou igual a 10, o seu valor será somado em 10 unidades, evitando que a div jogador saia da área da div fundogame.

O próximo passo é limitar a movimentação quando tecla S for pressionada.

3. Digite o código a seguir no local indicado pela Figura 46:

```
if (topo>=434) {  
  
    $("#jogador").css("top",topo-10);  
  
}
```

```
if (jogo.pressionou(TECLA.S)) {  
  
    var topo = parseInt($("#jogador").css("top"));  
    $("#jogador").css("top", topo+10);  
}
```

```
if (topo>=434) {  
  
    $("#jogador").css("top", topo-10);  
}  
}
```

## **Figura 46**

No código anterior, caso o valor da variável topo seja maior ou igual a 434, o seu valor será subtraído em 10 unidades, evitando que a div jogador saia da área da div fundoGame.

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html pelo Google Chrome.

Observe que a movimentação do jogador está limitada dentro da div fundogame (Figura 47).



**Figura 47**

## Animando a div inimigo1

Prosseguindo com o jogo, faremos com que a div inimigo1 seja exibida em uma posição aleatória (randômica) na propriedade top (eixo Y) e criaremos uma função que fará com que a div se movimente pelo eixo X, subtraindo o valor da propriedade left.

Inicialmente vamos criar uma variável com o nome de velocidade, que conterá o valor da velocidade do jogo e uma variável com o nome de posicaoY, que terá um valor randômico (aleatório) entre 0 e 334, fazendo com que a div inimigo1 seja posicionada aleatoriamente no eixo Y.

1. Digite o código a seguir no local indicado pela Figura 48.

```
var velocidade=5;
```

```
var posicaoY = parseInt(Math.random() * 334);
```

```
//Principais variáveis do jogo
```

```
var jogo = {}
```

```
var velocidade=5;
```

```
var posicaoY = parseInt(Math.random() * 334);
```

```
var TECLA = {
```

```
W: 87,
```

```
S: 83,
```

```
D: 68
```

```
}
```

## **Figura 48**

O próximo passo é chamar a função moveinimigo1 no game loop do jogo.

2. Digite o código a seguir no local indicado na Figura 49.

```
moveinimigo1();
```

```
function loop() {  
    movefundo();  
    movejogador();  
    moveinimigo1();  
}  
// Fim da função loop()
```

## **Figura 49**

Agora vamos criar a função moveinimigo1.

3. Digite o código a seguir no final da função start, como indicado na Figura 50.

```
function moveinimigo1() {  
  
    posicaoX = parseInt($("#inimigo1").css("left"));  
  
    $("#inimigo1").css("left",posicaoX-velocidade);  
  
    $("#inimigo1").css("top",posicaoY);  
  
    if (posicaoX<=0) {  
  
        posicaoY = parseInt(Math.random() * 334);  
        $("#inimigo1").css("left",694);  
        $("#inimigo1").css("top",posicaoY);  
    }  
}
```

```
} } //Fim da função moveinimigo1()
```

```
} // fim da função movejogador()
```

```
function moveinimigo1() {  
  
    posicaoX = parseInt($("#inimigo1").css("left"));  
    $("#inimigo1").css("left",posicaoX-velocidade);  
    $("#inimigo1").css("top",posicaoY);  
  
    if (posicaoX<=0) {  
  
        posicaoY = parseInt(Math.random() * 334);  
        $("#inimigo1").css("left",694);  
        $("#inimigo1").css("top",posicaoY);  
  
    }  
} //Fim da função moveinimigo1()
```

```
} // Fim da função start
```

## Figura 50

Criamos uma variável com o nome posicaoX que irá armazenar o valor atual da propriedade left da div inimigo1.

```
posicaoX = parseInt($("#inimigo1").css("left"));
```

Na linha a seguir estamos atualizando o valor da propriedade left com o valor atual da variável posicaoX menos o valor da variável velocidade, criando a animação.

```
$("#inimigo1").css("left",posicaoX-velocidade);
```

Na linha a seguir, para que a div inimigo1 apareça em um valor aleatório no eixo Y, estamos atualizando a propriedade top da div com o conteúdo da variável posicaoY.

```
$("#inimigo1").css("top",posicaoY);
```

A seguir estamos limitando a movimentação da div. Quando o valor da variável posicaoX for menor que 0, a div é reposicionada do lado direito da tela novamente:

```
$("#inimigo1").css("left",694);
```

Observe que a div é reposicionada em um valor aleatório no eixo Y novamente:

```
$("#inimigo1").css("top",posicaoY);
```

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html pelo Google Chrome.

Observe que agora o inimigo1 se movimentará pelo eixo X (Figura 51).



**Figura 51**

## Capítulo 9

### Criando a movimentação das demais divs do jogo

Neste capítulo criaremos o código que movimentará a div inimigo2 e a div amigo.

Vamos começar pela div inimigo2.

1. Abra o arquivo js.js.

Inicialmente chamaremos a função moveinimigo2 dentro do game loop.

2. Digite o código a seguir no local indicado na Figura 52.

```
moveinimigo2();
```

```
function loop() {  
    // Códigos de movimento  
    movefundo();  
    movejogador();  
    moveinimigo1();  
    moveinimigo2();  
}  
// Fim da função loop()
```

## **Figura 52**

3. No final da função start, vamos criar a função moveinimigo2. Digite o código a seguir no local indicado pela Figura 53.

```
function moveinimigo2() {  
  
    posicaoX = parseInt($("#inimigo2").css("left"));  
    $("#inimigo2").css("left",posicaoX-3);  
  
    if (posicaoX<=0) {  
  
        $("#inimigo2").css("left",775);  
  
    }  
  
} // Fim da função moveinimigo2()
```

```
    }
} // Fim da função moveinimigo1()
```

```
function moveinimigo2() {
    posicaoX = parseInt($("#inimigo2").css("left"));
    $("#inimigo2").css("left",posicaoX-3);

    if (posicaoX<=0) {
        $("#inimigo2").css("left",775);
    }
} // Fim da função moveinimigo2()
```

```
} // Fim da função start
```

## **Figura 53**

Na função anterior criamos uma variável com o nome posicaoX que armazenará o valor atual da propriedade left da div inimigo2.

```
posicaoX = parseInt($("#inimigo2").css("left"));
```

Na linha a seguir estamos atualizando o valor da propriedade left com o valor atual da variável posicaoX menos três unidades (-3), criando a animação.

```
$("#inimigo2").css("left",posicaoX-3);
```

A seguir estamos limitando a movimentação da div. Quando o valor da variável posicaoX for menor que 0, a div inimigo2 é reposicionada do lado direito da tela novamente:

```
$("#inimigo2").css("left",775);
```

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html pelo Google Chrome.

Observe que a animação da div inimigo2 no eixo X foi realizada (Figura 54).





## **Figura 54**

O próximo passo é criar a animação da div amigo. Faremos a animação da esquerda para a direita no eixo X.

6. Inicialmente vamos chamar a função moveamigo no game loop do jogo. Digite o código a seguir no local indicado pela Figura 55.

```
moveamigo();
```

```
function loop() {  
    movefundo();  
    movejogador();  
    moveinimigo1();  
    moveinimigo2();  
    moveamigo();  
}  
// Fim da função loop()
```

## **Figura 55**

7. Agora vamos criar a função moveamigo. Digite o código a seguir no local indicado na Figura 56:

```
function moveamigo() {  
  
    posicaoX = parseInt($("#amigo").css("left"));  
    $("#amigo").css("left",posicaoX+1);  
  
    if (posicaoX>906) {  
  
        $("#amigo").css("left",0);  
  
    }  
  
} // fim da função moveamigo()
```

```
    }  
}  
} // Fim da função moveinimigo2()
```

```
function moveamigo() {  
  
    posicaoX = parseInt($("#amigo").css("left"));  
    $("#amigo").css("left",posicaoX+1);  
  
    if (posicaoX>906) {  
  
        $("#amigo").css("left",0);  
  
    }  
} // fim da função moveamigo()  
  
}  
} // Fim da função start
```

## Figura 56

Na função anterior criamos uma variável com o nome posicaoX, que armazenará o valor atual da propriedade left da div amigo.

```
posicaoX = parseInt($("#amigo").css("left"));
```

Na linha a seguir estamos atualizando o valor da propriedade left com o valor atual da variável posicaoX mais uma unidade, criando a animação.

```
$("#amigo").css("left",posicaoX+1);
```

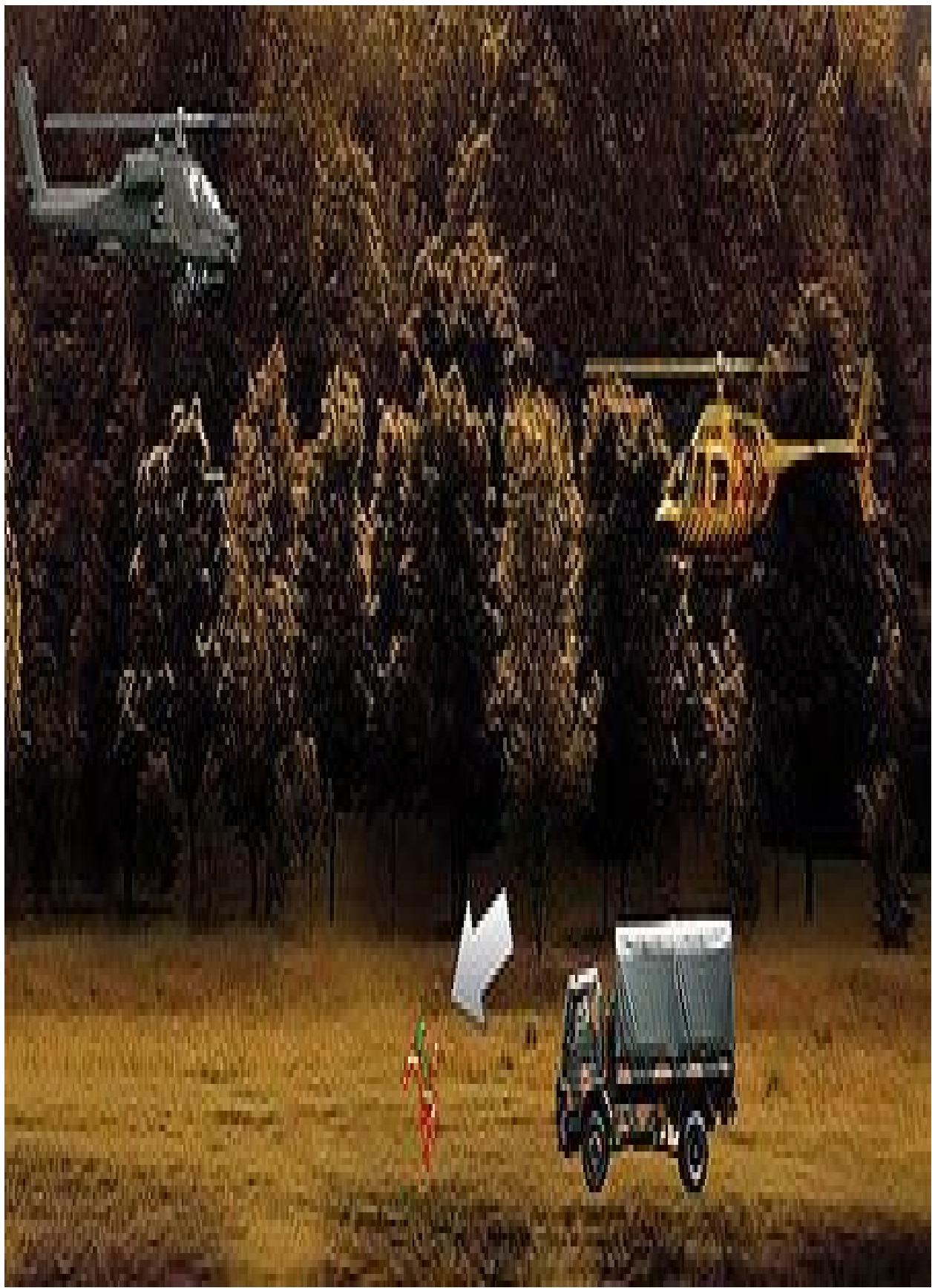
A seguir estamos limitando a movimentação da div. Quando o valor da variável posicaoX for maior que 906, a div é reposicionada do lado esquerdo da tela novamente:

```
$("#amigo").css("left",0);
```

8. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
9. Execute o arquivo index.html pelo Google Chrome.

Observe que a animação da div amigo no eixo X foi realizada (Figura 57).





**Figura 57**

## Capítulo 10

### Criando a função disparo

Neste capítulo criaremos a função que realizará os disparos do jogador. Esses disparos serão exibidos em uma div com o nome de disparo, que será criada pelo código JavaScript. Antes de fazer o código que vai realizar essa operação, vamos criar o código CSS que terá a formatação inicial da div disparo.

1. Abra o arquivo estilos.css.
2. Digite o código a seguir no final do arquivo.

```
#disparo {  
    width: 50px;  
    height: 8px;  
    position: absolute;  
    background-image: url(..../imgs/disparo.png);  
}
```

3. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar uma nova variável com o nome de podeAtirar, que irá controlar quando o jogador poderá realizar disparos ou não.

4. Abra o arquivo js.js.

5. Digite o código a seguir no local indicado na Figura 58.

```
var podeAtirar=true;
```

1000

## **Figura 58**

O próximo passo é chamar a função disparo toda vez que a tecla D for pressionada.

6. Digite o código a seguir no local indicado na Figura 59:

disparo();

```
if (jogo.pressionou(TECLA_D)) {  
    // move o jogador para a direita  
    movejogador();  
    // chama função Disparo  
    disparo();  
}  
// fim da função movejogador()
```

## Figura 59

7. A seguir vamos criar a função disparo. Digite o código a seguir no local indicado na Figura 60.

```
function disparo() {  
  
    if (podeAtirar==true) {  
  
        podeAtirar=false;  
  
        topo = parseInt($("#jogador").css("top"))  
        posicaoX= parseInt($("#jogador").css("left"))  
        tiroX = posicaoX + 190;  
        topoTiro=topo+37;  
        $("#fundoGame").append("<div id='disparo'></div>");  
        $("#disparo").css("top",topoTiro);  
        $("#disparo").css("left",tiroX);  
  
        var tempoDisparo=window.setInterval(executaDisparo, 30);  
    }  
}
```

```
    } //Fecha podeAtirar

    function executaDisparo() {
        posicaoX = parseInt($("#disparo").css("left"));
        $("#disparo").css("left", posicaoX+15);

        if (posicaoX>900) {
            window.clearInterval(tempoDisparo);
            tempoDisparo=null;
            $("#disparo").remove();
            podeAtirar=true;
        }
    } // Fecha executaDisparo()
} // Fecha disparo()
```

```
posicaoX = parseInt($("#amigo").css("left"));
$("#amigo").css("left",posicaoX+1);

if (posicaoX>906) {

    $("#amigo").css("left",0);

}
} // fim da função moveamigo()
```

```
function disparo() {

if (podeAtirar==true) {

    podeAtirar=false;

topo = parseInt($("#jogador").css("top"));
posicaoX= parseInt($("#jogador").css("left"))
tiroX = posicaoX + 190;
topoTiro=topo+37;
$("#fundoGame").append("<div id='disparo'></div>");
$("#disparo").css("top",topoTiro);
$("#disparo").css("left",tiroX);

var tempoDisparo=window.setInterval(executaDisparo, 30);

} //Fecha podeAtirar

function executaDisparo() {
    posicaoX = parseInt($("#disparo").css("left"));
    $("#disparo").css("left",posicaoX+15);

    if (posicaoX>900) {

        window.clearInterval(tempoDisparo);
        tempoDisparo=null;
        $("#disparo").remove();
        podeAtirar=true;

    }
} // Fecha executaDisparo()
} // Fecha disparo()

} // Fim da função start()
```

## Figura 60

Na função disparo realizamos as seguintes operações.

Inicialmente verificamos se o valor da variável podeAtirar é igual a true. Caso seja true, alteramos o valor da variável podeAtirar para false, para evitar que o jogador realize outro disparo.

```
function disparo() {  
  
    if (podeAtirar==true) {  
  
        podeAtirar=false;
```

Após esse procedimento armazenamos na variável topo a posição atual no eixo Y da div jogador e na variável posicaoX a posição atual da div jogador no eixo X. A seguir criamos uma variável com o nome de tiroX com valor da variável posicaoX +190 unidades e outra variável com o nome de topoTiro, que terá o valor da variável topo +37 unidades.

```
topo = parseInt($("#jogador").css("top"))  
  
posicaoX= parseInt($("#jogador").css("left"))  
  
tiroX = posicaoX + 190;
```

```
topoTiro=topo+37;
```

Dessa forma, temos o valor exato do local inicial do disparo (Figura 61).



tiroX  
topoTiro

## Figura 61

Na linha abaixo do código, criamos uma nova div com o nome de disparo:

```
$("#fundoGame").append("<div id='disparo'></div>");
```

E a posicionamos no seu local inicial:

```
$("#disparo").css("top",topoTiro);
```

```
$("#disparo").css("left",tiroX);
```

Após posicionar a div disparo à frente da div jogador, é criada uma função de tempo que fará a animação da div disparo:

```
var tempoDisparo=window.setInterval(executaDisparo, 30);
```

Na função tempoDisparo é criada uma animação onde a propriedade left da div disparo é somada em quinze unidades, realizando a animação:

```
posicaoX = parseInt($("#disparo").css("left"));
```

```
$("#disparo").css("left",posicaoX+15);
```

Quando a div disparo chegar no final da div fundoGame a função de tempo é parada para que a animação não seja mais realizada. A div disparo é removida pelo código `$("#disparo").remove();`

```
if (posicaoX>900) {
```

```
    window.clearInterval(tempoDisparo);
```

```
    tempoDisparo=null;
```

```
    $("#disparo").remove();
```

```
    podeAtirar=true;
```

Observe que o valor da variável `podeAtirar` é alterado para `true`, possibilitando assim que o jogador possa realizar um novo disparo.

8. Pressione as teclas `Ctrl + S` para salvar as alterações no arquivo.

9. Execute o arquivo `index.html` pelo Google Chrome.

10. Observe que, ao pressionar a tecla D, o disparo do jogador é realizado. Observe também que um novo disparo só poderá ser realizado quando a div disparo chegar ao final da div fundogame.



## **Figura 62**

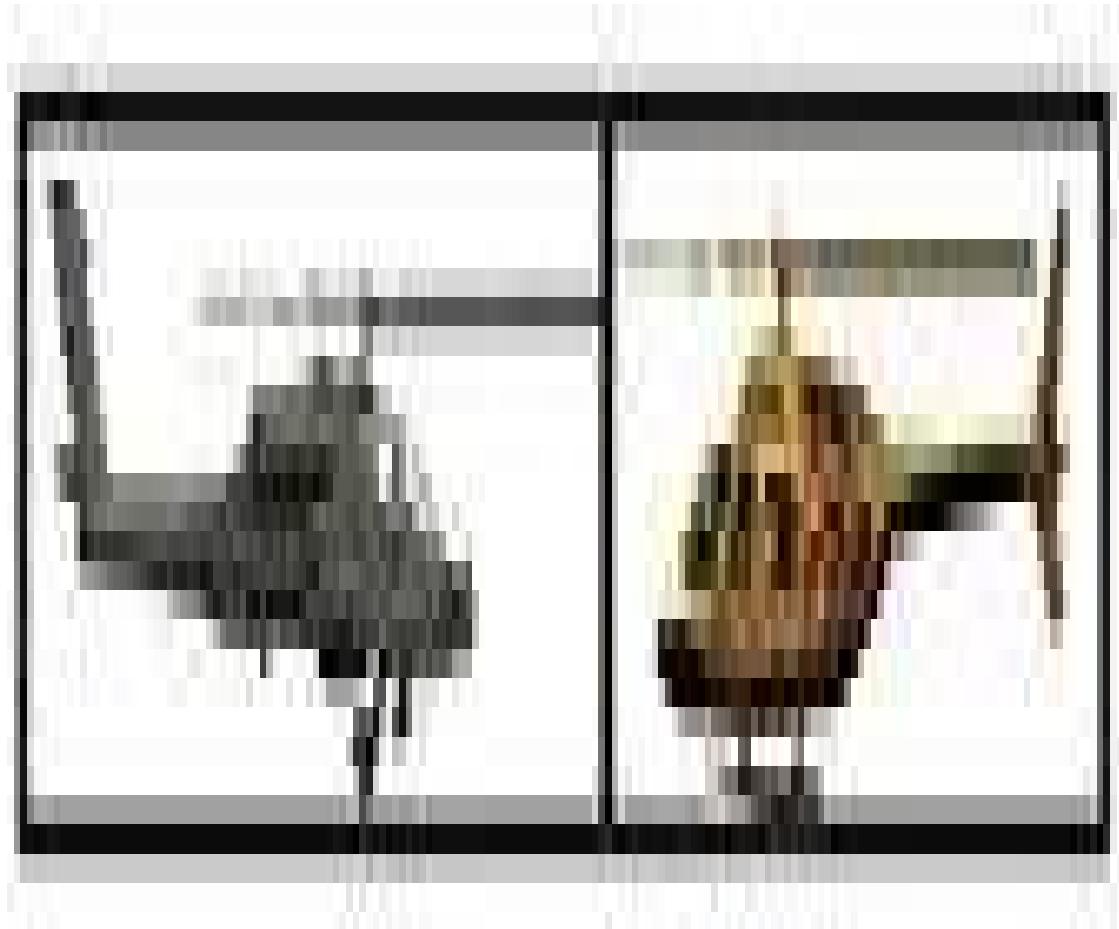
Com os disparos realizados e as principais animações prontas, o próximo passo é identificar as colisões do jogo.

## **Capítulo 11**

### **Colisões parte I**

A detecção de colisões em jogos nada mais é do que identificar a interseção de dois objetos diferentes. Esta é uma das funções mais importantes no desenvolvimento de jogos, pois são utilizadas para identificar a colisão e chamar uma resposta a essa colisão.

Neste jogo em desenvolvimento detectaremos a colisão entre uma div e outra. Veja o exemplo da Figura 63.



## **Figura 63**

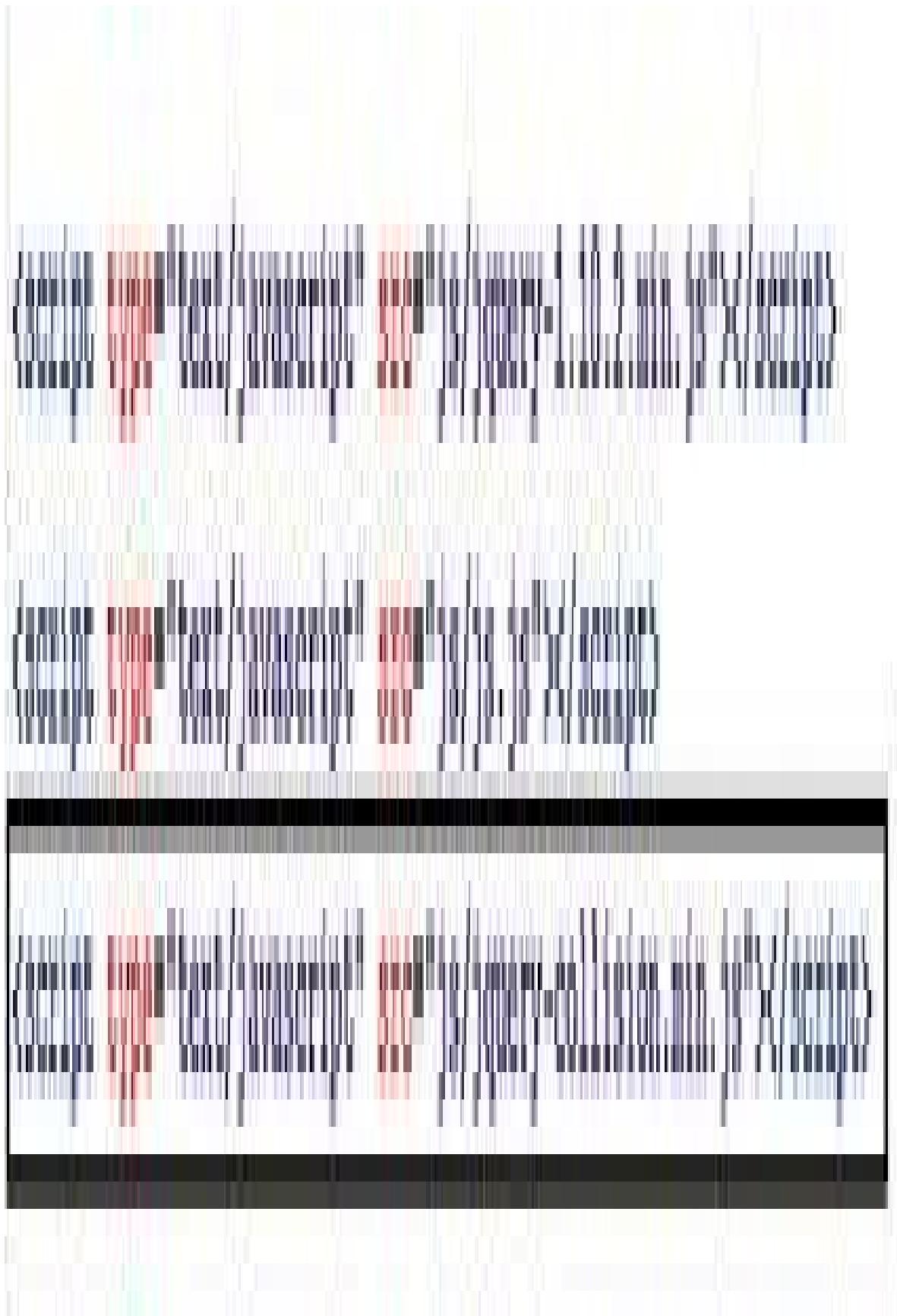
Para facilitar esse procedimento utilizaremos a biblioteca jQuery Collision.

Não será necessário realizar o download do arquivo jquery-collision.min.js, pois o arquivo já está disponível na sua pasta js.

Inicialmente vamos vincular o jQuery Collision ao arquivo index.html.

1. Abra o arquivo index.html.
2. Digite o código a seguir no local indicado na Figura 64:

```
<script type="text/javascript" src="js/jquery-collision.min.js"></script>
```



## **Figura 64**

3. Pressione as teclas Ctrl + S para salvar as alterações.

O próximo passo é chamar uma função com o nome de colisao no game loop do jogo.

4. No arquivo js.js digite o código indicado na Figura 65:

```
function loop() {  
    movefundo();  
    movejogador();  
    moveinimigo1();  
    moveinimigo2();  
    movebulho();  
    colisao();  
}  
// Fim da função loop()
```

## **Figura 65**

O próximo passo é criar a função colisao.

5. Digite o código a seguir no local indicado na Figura 66.

```
function colisao() {  
  
    var colisao1 = ($("#jogador").collision($("#inimigo1")));  
    // jogador com o inimigo1  
    if (colisao1.length>0) {  
  
        posicaoY = parseInt(Math.random() * 334);  
        $("#inimigo1").css("left",694);  
        $("#inimigo1").css("top",posicaoY);  
    }  
  
} //Fim da função colisao()
```

```
} // Fecha disparo()
```

```
function colisao() {  
  
    var colisao1 = ($("#jogador").collision($("#inimigo1")));  
    // jogador com o inimigo1  
    if (colisao1.length>0) {  
  
        posicaoY = parseInt(Math.random() * 334);  
        $("#inimigo1").css("left", 694);  
        $("#inimigo1").css("top", posicaoY);  
    }  
  
} //Fim da função colisao()
```

```
} // Fim da função start
```

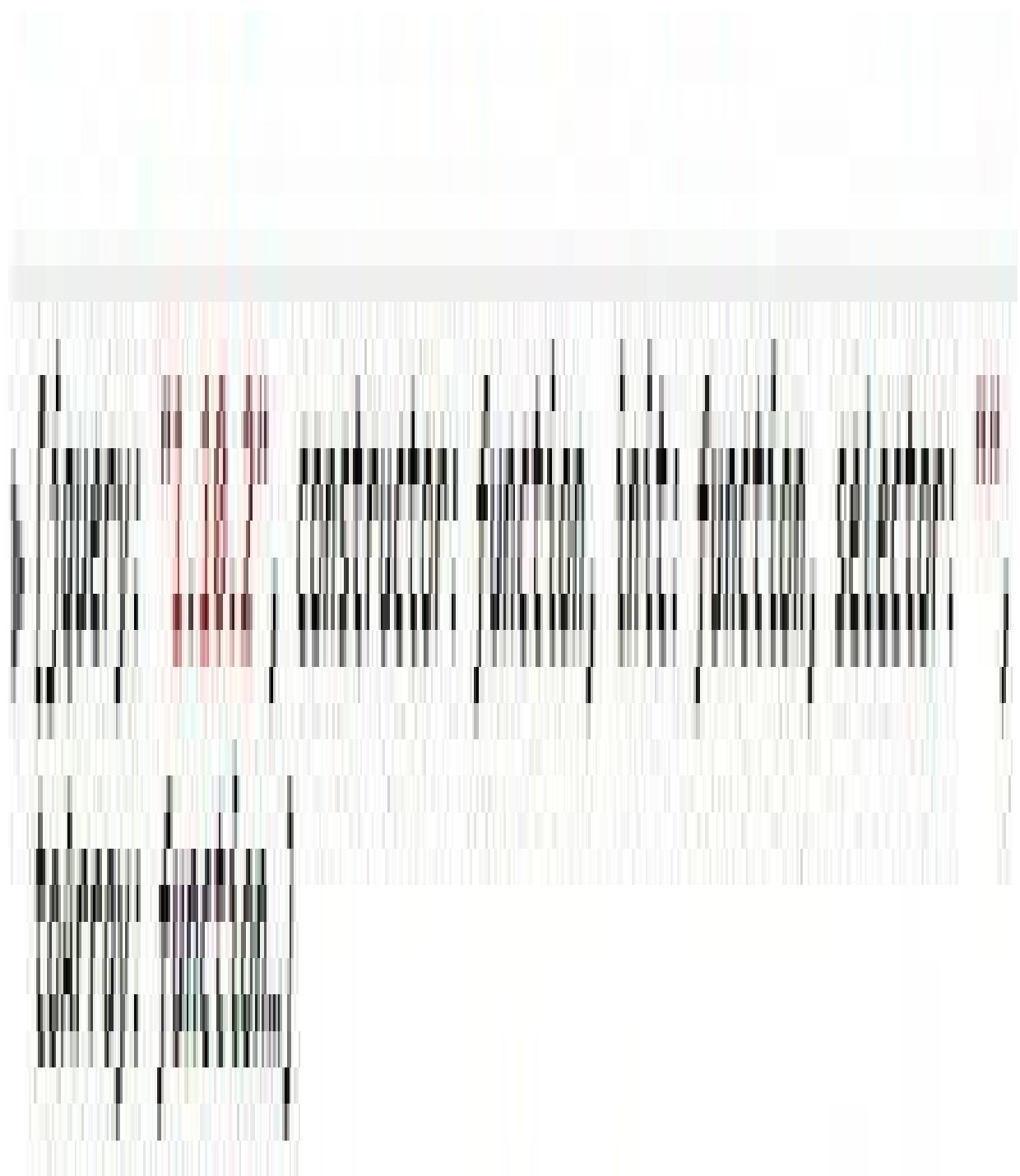
## **Figura 66**

Na função anterior observe que criamos uma variável com o nome de colisao1, que identificará a colisão da div jogador com a div inimigo1.

```
var colisao1 = ($("#jogador").collision($("#inimigo1")));
```

A função collision pertence à biblioteca jQuery Collision.

Caso haja a colisão da div jogador com a div inimigo1, o conteúdo da variável colisao1 receberá as informações da colisão. Esse conteúdo será armazenado na variável em forma de array. Veja o exemplo na Figura 67 do conteúdo armazenado na array<sup>1</sup>.



## Figura 67

Essas informações não serão necessárias para o jogo, queremos apenas saber se houve a colisão ou não; assim, iremos verificar o tamanho desta variável pela propriedade length. Caso a variável tenha valor maior que 0, houve a colisão:

```
if (colisao1.length>0) {
```

Com a colisão detectada, inicialmente iremos somente reposicionar a div inimigo1:

```
posicaoY = parseInt(Math.random() * 334);  
$("#inimigo1").css("left",694);  
$("#inimigo1").css("top",posicaoY);
```

O próximo passo é criar a função que exibirá uma explosão quando a colisão da div jogador com a div inimigo1 for detectada.

Inicialmente criaremos duas variáveis que irão capturar a posição atual da div inimigo1 para que a explosão seja exibida exatamente onde a colisão foi detectada.

Após isso chamaremos uma função com o nome de explosao1.

6. Digite o código a seguir no local indicado pela Figura 68. Veja que o código será digitado no local onde a colisão da div jogador com a div inimigo1 é detectada:

```
inimigo1X = parseInt($("#inimigo1").css("left"));
```

```
inimigo1Y = parseInt($("#inimigo1").css("top"));
```

```
explosao1(inimigo1X,inimigo1Y);
```

```
function colisao() {  
  
    var colisao1 = ($("#jogador").collision($("#inimigo1")));  
    // jogador com o inimigo1  
    if (colisao1.length>0) {  
  
        inimigo1X = parseInt($("#inimigo1").css("left"));  
        inimigo1Y = parseInt($("#inimigo1").css("top"));  
        explosao1(inimigo1X,inimigo1Y);  
  
        posicaoY = parseInt(Math.random() * 334);  
        $("#inimigo1").css("left",694);  
        $("#inimigo1").css("top",posicaoY);  
    }  
  
} //Fim da função colisao()
```

## **Figura 68**

O próximo passo é criar a função explosao1. Digite o código a seguir no local indicado pela Figura 69.

```
//Explosão 1

function explosao1(inimigo1X,inimigo1Y) {

    somExplosao.play();

    $("#fundoGame").append("<div id='explosao1'></div>");

    $("#explosao1").css("background-image", "url(imgs/explosao.png)");

    var div=$("#explosao1");

    div.css("top", inimigo1Y);

    div.css("left", inimigo1X);

    div.animate({width:200, opacity:0}, "slow");

    var tempoExplosao=window.setInterval(removeExplosao, 1000);

}

function removeExplosao() {

    div.remove();

}
```

```
window.clearInterval(tempoExplosao);
```

```
tempoExplosao=null;
```

```
}
```

```
} // Fim da função explosao1()
```

```
 } //Fim da função colisao()
```

```
//Explosão 1
function explosao1(inimigo1X,inimigo1Y) {
    somExplosao.play();
    $("#fundoGame").append("<div id='explosao1'></div>");
    $("#explosao1").css("background-image", "url(imgs/explosao.png)");
    var div=$("#explosao1");
    div.css("top", inimigo1Y);
    div.css("left:", inimigo1X);
    div.animate({width:200, opacity:0}, "slow");

    var tempoExplosao=window.setInterval(removeExplosao, 1000);

    function removeExplosao() {
        div.remove();
        window.clearInterval(tempoExplosao);
        tempoExplosao=null;
    }
} // Fim da função explosao1()
```

```
) // Fim da função start
```

**Figura 69**



Na função explosao1, observe que inicialmente criamos a div explosao1 dentro da div fundoGame e indicamos a imagem de fundo que será utilizada.

```
$("#fundoGame").append("<div id='explosao1'></div>");  
$("#explosao1").css("background-image", "url(imgs/explosao.png)");
```

Para facilitar a digitação criamos uma variável com o nome de div com o conteúdo \$("#explosao1").

A seguir posicionamos a div explosao1 no mesmo local onde houve a colisão:

```
div.css("top", inimigo1Y);  
div.css("left", inimigo1X);
```

Pela propriedade animate, criamos uma animação onde a div irá aumentar de tamanho até 200px de largura e perder a sua opacidade gradativamente até 0%:

```
div.animate({width:200, opacity:0}, "slow");
```

Observe que o valor slow foi utilizado para que a animação seja realizada lentamente.

A seguir criamos uma variável de tempo que removerá a div explosao1 após um segundo:

```
var tempoExplosao=window.setInterval(removeExplosao, 1000);
```

```
function removeExplosao() {
```

```
    div.remove();
```

```
    window.clearInterval(tempoExplosao);
```

```
    tempoExplosao=null;
```

```
}
```

```
} // Fim da função explosao1()
```

7. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar a formatação CSS inicial das divs explosao1 e explosao2.

8. Abra o arquivo estilos.css e digite o código a seguir no final do arquivo:

```
#explosao1 {  
    width:15px;  
    height:87px;  
    position:absolute;  
  
}  
  
#explosao2 {  
    width:15px;  
    height:87px;  
    position:absolute;  
  
}
```

9. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

10. Execute o arquivo index.html pelo Google Chrome.

11. Observe que a div explosao1 será exibida caso haja a colisão da div jogador com a div inimigo1 (Figura 70).



## **Figura 70**

1 Informação obtida através do comando console.log(colisao1). Pelo comando console.log o valor da variável é exibido na janela Developers tools do browser.

## Capítulo 12

### Colisões parte II

Neste capítulo daremos continuidade à detecção de colisões do jogo e à criação dos comportamentos que serão executados depois que cada colisão for detectada.

Inicialmente criaremos as variáveis que serão utilizadas para detectar as colisões entre os demais objetos do jogo.

1. Abra o arquivo js.js.
2. Digite o código a seguir dentro da função colisao, no local indicado na Figura 71.

```
var colisao2 = ($("#jogador").collision($("#inimigo2")));  
var colisao3 = ($("#disparo").collision($("#inimigo1")));  
var colisao4 = ($("#disparo").collision($("#inimigo2")));  
var colisao5 = ($("#jogador").collision($("#amigo")));  
var colisao6 = ($("#inimigo2").collision($("#amigo")));
```

```
function colisao() {  
    var colisao1 = ($("#jogador").collision($("#inimigo1")));  
    var colisao2 = ($("#jogador").collision($("#inimigo2")));  
    var colisao3 = ($("#disparo").collision($("#inimigo1")));  
    var colisao4 = ($("#disparo").collision($("#inimigo2")));  
    var colisao5 = ($("#jogador").collision($("#amigo")));  
    var colisao6 = ($("#inimigo2").collision($("#amigo")));  
}
```

## **Figura 71**

Vamos iniciar verificando a colisão da div jogador com a div inimigo2. Observe que para isso devemos verificar o tamanho da variável colisao2 pela propriedade length, como utilizado anteriormente.

A função descrita a seguir deve ser criada após o if que detecta o tamanho da variável colisao1, no local indicado na Figura 72.

```
// jogador com o inimigo1
if (colisaol.length>0) {

    inimigo1X = parseInt($("#inimigo1").css("left"));
    inimigo1Y = parseInt($("#inimigo1").css("top"));
    explosao1(inimigo1X,inimigo1Y);

    posicaoY = parseInt(Math.random() * 334);
    $("#inimigo1").css("left",694);
    $("#inimigo1").css("top",posicaoY);
}

//Fim da função colisao()

//Explosão 1
function explosao1(inimigo1X,inimigo1Y) {
```

**Figura 72**

Em caso de dúvida sobre o local correto para inserir o código, consulte o arquivo

3. Após posicionar o cursor no local correto, digite o código a seguir:

```
// jogador com o inimigo2
if (colisao2.length>0) {
    inimigo2X = parseInt($("#inimigo2").css("left"));
    inimigo2Y = parseInt($("#inimigo2").css("top"));
    explosao2(inimigo2X,inimigo2Y);
    $("#inimigo2").remove();
    reposicionaInimigo2();
}
```



Observe pelo código que, caso haja a colisão da div jogador com a div inimigo2, a posição atual da div inimigo2 será armazenada nas variáveis inimigo2X e inimigo2Y e a função explosao2 será chamada. Criaremos a função explosao2 mais à frente.

```
inimigo2X = parseInt($("#inimigo2").css("left"));
inimigo2Y = parseInt($("#inimigo2").css("top"));
explosao2(inimigo2X,inimigo2Y);
```

A seguir a div inimigo2 é removida e a função reposicionaInimigo2 é chamada.

```
$("#inimigo2").remove();
reposicionaInimigo2();
```

A função reposicionaInimigo2, que iremos criar mais à frente, terá a função de recriar e reposicionar a div inimigo2.

4. Pressione a tecla Enter e digite o código a seguir. Nesse código identificaremos a colisão da div disparo e a div inimigo1.

```
// Disparo com o inimigo1
```

```
if (colisao3.length>0) {  
  
    inimigo1X = parseInt($("#inimigo1").css("left"));  
    inimigo1Y = parseInt($("#inimigo1").css("top"));  
  
    explosao1(inimigo1X,inimigo1Y);  
    $("#disparo").css("left",950);  
  
    posicaoY = parseInt(Math.random() * 334);  
    $("#inimigo1").css("left",694);  
    $("#inimigo1").css("top",posicaoY);  
  
}
```



Observe pelo código que, caso haja a colisão da div disparo com a div inimigo1, a posição atual da div inimigo1 será armazenada nas variáveis inimigo1X e inimigo1Y e a função explosao1 será chamada.

Estamos reposicionando também a div disparo pelo código `$("#disparo").css("left",950)`. Isso fará com que a div disparo seja posicionada fora da área da div fundoGame. A função disparo irá detectar essa posição, excluindo a div e permitindo assim que o jogador possa efetuar outro disparo.

A seguir estamos reposicionando a div inimigo1 em uma posição randômica no eixo Y e na sua posição original do eixo X:

```
posicaoY = parseInt(Math.random() * 334);  
$("#inimigo1").css("left",694);  
$("#inimigo1").css("top",posicaoY);
```

5. O próximo passo é identificar a colisão da div disparo com a div inimigo2. Pressione a tecla Enter e digite o código a seguir:

```
// Disparo com o inimigo2
```

```
if (colisao4.length>0) {
```

```
inimigo2X = parseInt($("#inimigo2").css("left"));
inimigo2Y = parseInt($("#inimigo2").css("top"));
$("#inimigo2").remove();

explosao2(inimigo2X,inimigo2Y);
$("#disparo").css("left",950);

reposicionaInimigo2();
}
```

Observe que nesse código estamos também chamando a função explosao2 e a função reposicionaInimigo2, que irá recriar e reposicionar a div inimigo2 no jogo.

6. O próximo passo é identificar a colisão da div jogador com a div amigo. Pressione a tecla Enter e digite o código a seguir.

```
// jogador com o amigo
```

```
if (colisao5.length>0) {
```

```
reposicionaAmigo();
```

```
$("#amigo").remove();
```

```
}
```

Antes de testar o jogo precisamos criar as funções explosao2, reposicionaInimigo2 e reposicionaAmigo.

7. Posicione o cursor no local indicado na Figura 73.

```
var tempoExplosao=window.setInterval(removeExplosao, 1000);

function removeExplosao() {
    div.remove();
    window.clearInterval(tempoExplosao);
    tempoExplosao=null;
}

} // Fim da função explosao()

} // Fim da função start
```

### Figura 73

8. Pressione a tecla Enter e digite o código a seguir para criar a função explosao2.

```
//Explosão2
```

```
function explosao2(inimigo2X,inimigo2Y) {  
  
    $("#fundoGame").append("<div id='explosao2'></div>");  
    $("#explosao2").css("background-image", "url(imgs/explosao.png)");  
    var div2=$("#explosao2");  
    div2.css("top", inimigo2Y);  
    div2.css("left", inimigo2X);  
    div2.animate({width:200, opacity:0}, "slow");  
    var tempoExplosao2=window.setInterval(removeExplosao2, 1000);  
  
    function removeExplosao2() {  
  
        div2.remove();  
        window.clearInterval(tempoExplosao2);  
    }  
}
```

```
tempoExplosao2=null;  
  
}  
  
}  
} // Fim da função explosao2()
```

9. A seguir vamos criar a função reposicionaInimigo2. Pressione a tecla Enter e digite o código a seguir:

```
//Reposiciona Inimigo2  
  
function reposicionaInimigo2() {  
  
    var tempoColisao4=window.setInterval(reposiciona4, 5000);  
  
    function reposiciona4() {  
  
        window.clearInterval(tempoColisao4);  
  
        tempoColisao4=null;
```

```
if (fimdejogo==false) {  
  
    $("#fundoGame").append("<div id=inimigo2></div>");  
  
}  
  
}  
  
}
```

Na função reposicionaInimigo2 criamos uma função de tempo que recriará a div inimigo2 caso a variável fimdejogo seja igual a false. A variável fimdejogo evitará que essa div seja recriada caso o jogo esteja finalizado. Criaremos essa variável no final deste capítulo.

10. A seguir vamos criar a função reposicionaAmigo. Pressione a tecla Enter e digite o código a seguir:

```
//Reposiciona Amigo  
  
function reposicionaAmigo() {  
  
    var tempoAmigo=window.setInterval(reposiciona6, 6000);
```

```
function reposiciona6() {  
  
    window.clearInterval(tempoAmigo);  
  
    tempoAmigo=null;  
  
    if (fimdejogo==false) {  
  
        $("#fundoGame").append("<div id='amigo' class='anima3'></div>");  
  
    }  
  
}  
  
}  
  
} // Fim da função reposicionaAmigo()
```

Observe que nessa função a div amigo vinculada à classe anima3 será recriada após seis segundos caso a variável fimdejogo seja igual a false.

11. Para finalizar, vamos criar a variável fimdejogo. Digite o código a seguir no local indicado pela Figura 74.

```
var fimdejogo=false;
```

```
//Principais variaveis do jogo
```

```
var jogo = ()
```

```
var velocidade=5;
```

```
var posicaoY = parseInt(Math.random() * 334);
```

```
var podeAtirar=true;
```

```
var fimdejogo=false;
```

### **Figura 74**

12. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

13. Execute o arquivo index.html pelo Google Chrome.

Observe que as colisões indicadas foram detectadas, inclusive a colisão da div jogador com a div amigo (Figura 75).



## **Figura 75**

Falta apenas detectarmos a colisão da div inimigo2 com a div amigo. Criaremos o código dessa detecção no próximo capítulo, pois nessa colisão exibiremos uma nova animação.

## **Capítulo 13**

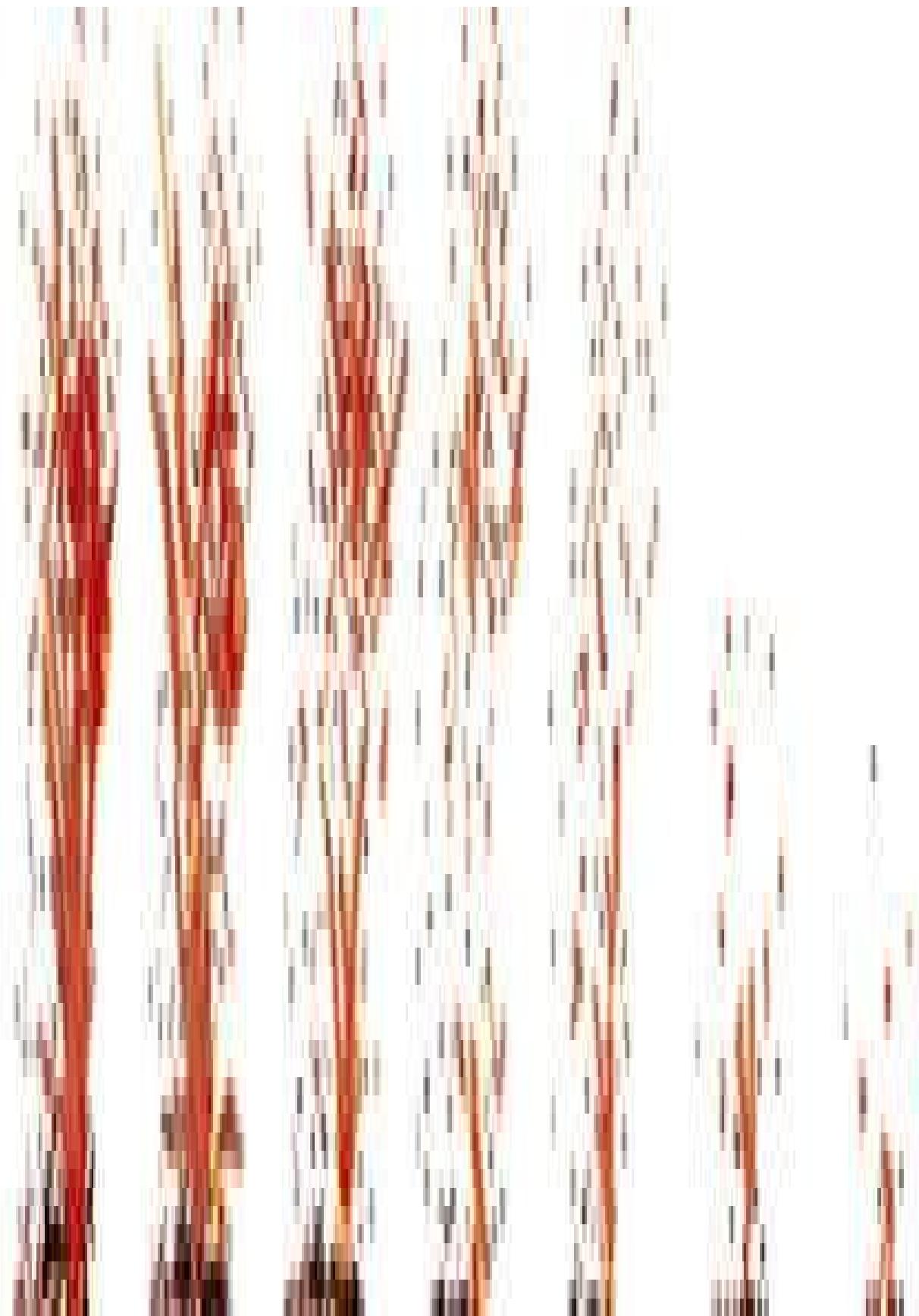
### **Colisões parte III**

Neste capítulo criaremos a explosão que será exibida caso a div inimigo2 entre em colisão com a div amigo (Figura 76).



## **Figura 76**

Inicialmente vamos criar uma animação CSS onde exibiremos a imagem amigo\_explosao.png (Figura 77).



## **Figura 77**

Observe que a imagem possui sete quadros-chave. Cada um desses quadros possui 44px de largura por 51px de altura.

1. Abra o arquivo estilos.css.
2. Digite o código a seguir no final do arquivo.

```
#explosao3 {
```

```
width:44px;
```

```
height:51px;
```

```
position:absolute;
```

```
}
```

```
.anima4 {
```

```
width:44px;
```

```
height:51px;
```

```
background-image:url(..../imgs/amigo_explorao.png);
```

```
animation:play3 .5s steps(7) infinite;  
-webkit-animation: play3 .5s steps(7) infinite;  
-moz-animation: play3 .5s steps(7) infinite;  
-ms-animation: play3 .5s steps(7) infinite;  
-o-animation: play3 .5s steps(7) infinite;  
}  
}
```

```
@keyframes play3 {  
from {background-position:0px;}  
to {background-position:-308px;}}
```

```
}
```

```
@-webkit-keyframes play3 {  
from { background-position: 0px; }  
to { background-position: -308px; }}
```

```
}
```

```
@-moz-keyframes play3 {
```

```
from { background-position: 0px; }

to { background-position: -308px; }

}
```

```
@-ms-keyframes play3 {

from { background-position: 0px; }

to { background-position: -308px; }

}
```

```
@-o-keyframes play3 {

from { background-position: 0px; }

to { background-position: -308px; }

}
```

3. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é chamar uma função que terá o nome de explosao3 quando o inimigo2 entrar em colisão com a div amigo.

4. Abra o arquivo js.js.

5. Digite o código a seguir no local indicado pela Figura 78.

```
//Inimigo2 com o amigo
```

```
if (colisao6.length>0) {
```

```
    amigoX = parseInt($("#amigo").css("left"));
```

```
    amigoY = parseInt($("#amigo").css("top"));
```

```
    explosao3(amigoX,amigoY);
```

```
    $("#amigo").remove();
```

```
    reposicionaAmigo();
```

```
}
```

```
// jogador com o amigo

if (colisao5.length>0) {

    reposicionaAmigo();
    $("#amigo").remove();

}

//Inimigo2 com o amigo

if (colisao6.length>0) {

    amigoX = parseInt($("#amigo").css("left"));
    amigoY = parseInt($("#amigo").css("top"));
    explosao3(amigoX,amigoY);
    $("#amigo").remove();

    reposicionaAmigo();

}

} //Fim da função colisao()
```

**Figura 78**



Quando a colisão da div inimigo2 com a div amigo é detectada, inicialmente armazenamos na variável amigoX e na variável amigoY a posição atual da div amigo e é chamada a função explosao3 para que seja exibida a explosão (a função explosao3 será criada a seguir). Veja que a div amigo é removida pelo código `$("#amigo").remove()` e a função reposicionaAmigo é chamada para recriar a div após seis segundos.

```
$("#amigo").remove();
```

```
reposicionaAmigo();
```

Agora vamos criar a função explosao3.

6. Digite o código a seguir no local indicado pela Figura 79.

```
//Explosão3
```

```
function explosao3(amigoX,amigoY) {  
    $("#fundoGame").append("<div id='explosao3' class='anima4'></div>");  
    $("#explosao3").css("top",amigoY);  
    $("#explosao3").css("left",amigoX);  
    var tempoExplosao3=window.setInterval(resetaExplosao3, 1000);  
    function resetaExplosao3() {
```

```
$("#explosao3").remove();  
window.clearInterval(tempoExplosao3);  
tempoExplosao3=null;  
  
}  
  
} // Fim da função explosao3
```

```
    } // Fim da função reposicionaAmigo()
```

```
//Explosao3
```

```
function explosao3(amigoX,amigoY) {
```

```
    $("#fundoGame").append("<div id='explosao3' class='animai'></div>");
```

```
    $("#explosao3").css("top",amigoY);
```

```
    $("#explosao3").css("left",amigoX);
```

```
    var tempoExplosao3=window.setInterval(resetaExplosao3, 1000);
```

```
    function resetaExplosao3() {
```

```
        $("#explosao3").remove();
```

```
        window.clearInterval(tempoExplosao3);
```

```
        tempoExplosao3=null;
```

```
}
```

```
    } // Fim da função explosao3
```

```
} // Fim da função start
```

**Figura 79**



A diferença nesta função é que criamos uma nova div com o nome de explosao3 dentro da div fundoGame utilizando a classe anima4.

```
$("#fundoGame").append("<div id='explosao3' class='anima4'></div>");
```

Essa animação será exibida por um segundo até que a div amigo seja reposicionada novamente.

Vamos testar o código.

7. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
8. Execute o arquivo index.html pelo Google Chrome.

Observe que, quando a div inimigo2 entra em colisão com div amigo, a div explosao3 será exibida (Figura 80).



**Figura 80**

## Capítulo 14

### Pontuação

Chegou a hora de inserirmos a pontuação do jogo. Nesse jogo iremos contar a quantidade de amigos salvos e a quantidade de amigos perdidos. Cada helicóptero abatido valerá 100 pontos e cada caminhão abatido valerá 50 pontos.

Para criar a pontuação inicialmente será necessário criar as variáveis que receberão a pontuação do jogador. Cada variável de pontuação será somada em suas respectivas colisões – por exemplo, quando a div disparo entrar em colisão com a div inimigo1 a variável de pontuação do jogador será somada em 100 unidades, e assim por diante.

Antes de tudo é preciso criar uma div dentro da div fundoGame para que seja exibida a pontuação.

1. Abra o arquivo js.js.
2. Digite o código a seguir no local indicado na Figura 81.

```
$("#fundoGame").append("<div id='placar'></div>");
```

```
    $({{>}}<div>{{>}}<div>{{>}}
```

## **Figura 81**

3. Pressione a teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar a formatação CSS para essa div e para a tag <h2>.

4. Abra o arquivo estilos.css.

5. Digite o código a seguir no final do arquivo.

```
#placar {  
    width: 450px;  
    height: 50px;  
    position: absolute;  
    left: 5px;  
    top: 590px;  
  
}  
  
h2 {  
    font-family: Titulo;
```

```
font-size:20px;
```

```
color:#FFF
```

```
}
```

6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Com a formatação criada, a seguir vamos criar as variáveis.

7. No arquivo js.js, digite o código a seguir no local indicado pela Figura 82.

```
var pontos=0;
```

```
var salvos=0;
```

```
var perdidos=0;
```

//Principais variáveis do jogo

var jogo = {};

var velocidade=5;

var posicaoY = parseInt(Math.random() \* 334);

var podeAtirar=true;

var fimdejogo=false;

var pontos=0;

var salvos=0;

var perdidos=0;

## **Figura 82**

Vamos iniciar somando a variável pontos toda vez que a div disparo entrar em colisão com a div inimigo1. Lembre que quando o jogador acertar o helicóptero ele ganhará 100 pontos.

8. Digite o código a seguir no local indicado pela Figura 83.

```
pontos=pontos+100;
```

```
if (col1sos.length > 0) {  
    pontos = pontos + 100;  
    int inicioX = parseInt($('input[id^="inicio"]').css('left'));  
    int inicioY = parseInt($('input[id^="inicio"]').css('top'));  
    int fimX = parseInt($('input[id^="fim"]').css('left'));  
    int fimY = parseInt($('input[id^="fim"]').css('top'));
```

### **Figura 83**

O próximo passo é somar em cinquenta unidades toda vez que a div dispero entrar em colisão com div inimigo2.

9. Digite o código a seguir no local indicado pela Figura 84.

```
pontos=pontos+50;
```

```
if (colisao.length>0) {  
    if (colisao[0].y <= 100) {  
        pontos=pontos+50;  
    }  
    if (colisao[0].y >= 900) {  
        pontos=pontos-50;  
    }  
}  
  
minigo2X = parseInt($('div[minigo2]').css('left'));  
  
minigo2Y = parseInt($('div[minigo2]').css('top'));
```

### **Figura 84**

Seguindo esse mesmo princípio, somaremos em uma unidade o valor da variável salvos toda vez que a div jogador entrar em colisão com a div amigo.

10. Digite o código a seguir no local indicado pela Figura 85.

```
salvos++;
```



## **Figura 85**

Para finalizar, somaremos o valor da variável perdidos toda vez que a div inimigo2 entrar em colisão com a div amigo.

11. Digite o código a seguir no local indicado pela Figura 86.

```
perdidos++;
```

```
if (colisao6.length>0) {  
    perdidos++;  
    amigoX = parseInt($("#amigo").css("left"));  
    amigoY = parseInt($("#amigo").css("top"));  
    explosao3(amigoX,amigoY);  
}
```

## **Figura 86**

Para que não seja necessário atualizar o conteúdo da div placar toda vez que uma variável for atualizada, criaremos uma função que será chamada no game loop do jogo que irá atualizar valores das variáveis exibidas na div placar.

12. Digite o código a seguir no local indicado pela Figura 87.

```
placar();
```

```
function loop() {  
    // Códigos de movimento  
    movefundo();  
    movejogador();  
    moveinimigo1();  
    moveinimigo2();  
    moveamigo();  
    colisao();  
    placar();  
}  
// Fim da função loop()
```

## **Figura 87**

13. Agora vamos criar a função placar. Digite o código a seguir no final do arquivo, mas dentro da função start, como indicado na Figura 88.

```
function placar() {  
  
    $("#placar").html("<h2> Pontos: " + pontos + " Salvos: " + salvos + " Perdidos:  
    " + perdidos + "</h2>");  
  
} //fim da função placar()
```

1000

## REFERENCES

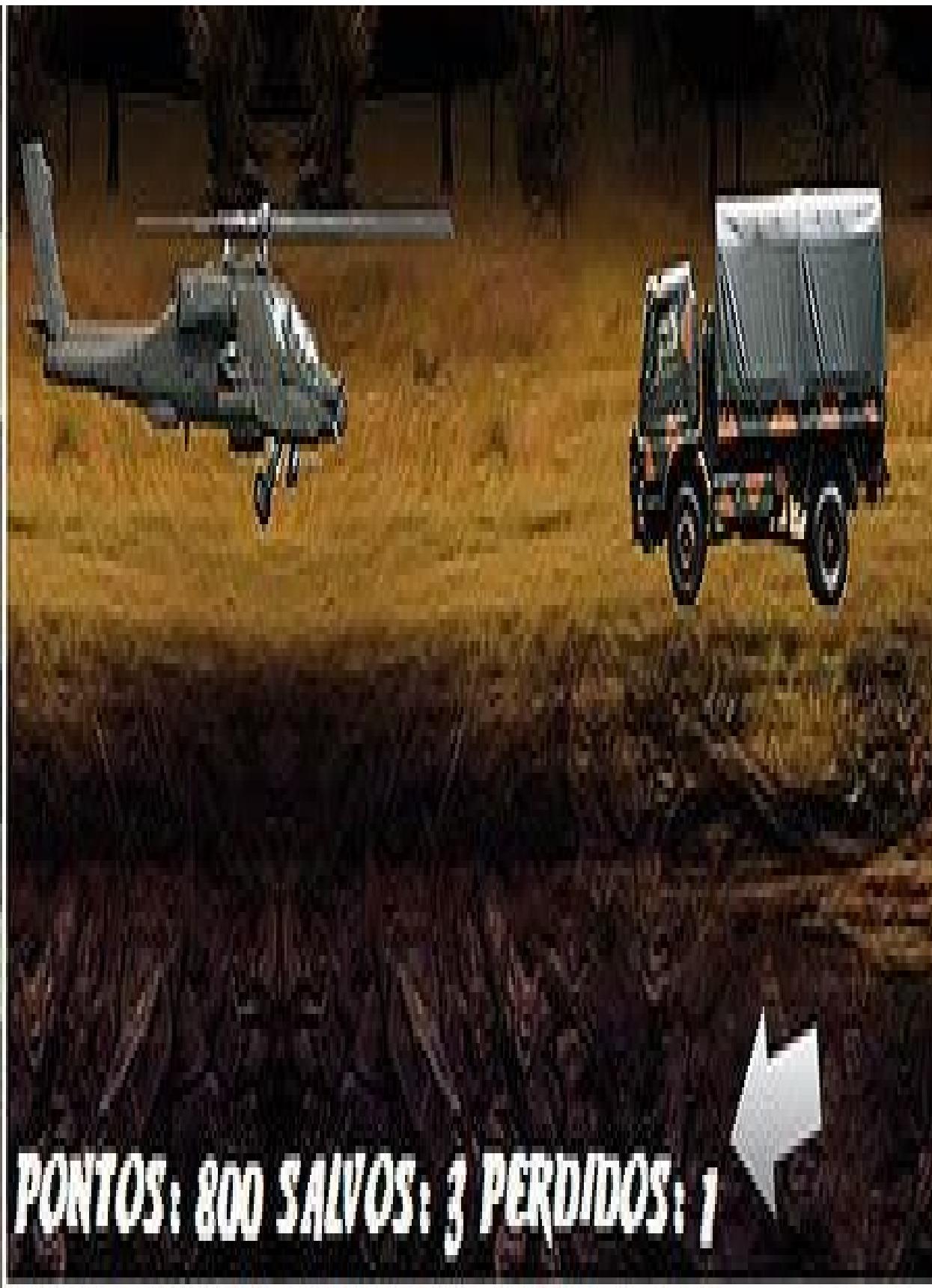
## **Figura 88**

Observe que essa função vai atualizar a div placar com o valor de cada uma das variáveis de pontuação que são utilizadas no jogo.

14. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

15. Execute o arquivo index.html pelo Google Chrome.

Observe que a pontuação do jogador é exibida na div placar (Figura 89).



**PONTOS: 800 SALVOS, 3 PERDIDOS.**

**Figura 89**

## Capítulo 15

### Criando uma barra de energia

A vida do jogador está muito fácil. Os inimigos podem atingi-lo e nada acontece. Para acabar com essa facilidade, vamos criar uma barra de energia para o jogador. Caso ele seja atingindo três vezes, será o fim do jogo. Mas antes de colocar o tão famoso “game over” criaremos a função que exibirá a barra de energia.

Iniciaremos criando a div onde essa barra será exibida.

1. Abra o arquivo js.js.
2. Digite o código a seguir no local indicado na Figura 90.

```
$("#fundoGame").append("<div id='energia'></div>");
```

```
S("body").append();
```

```
S("body").append("<div id='parent' class='parent>");
```

```
S("body").append("<div id='child' class='child>");
```

```
S("body").append("<div id='parent' class='parent>");
```

```
S("body").append("<div id='child' class='child>");
```

```
S("body").append("<div id='parent' class='parent>");
```

```
S("body").append("<div id='child' class='child>");
```

## **Figura 90**

3. Pressione a teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar a formatação CSS para essa div.

4. Abra o arquivo estilos.css.

5. Digite o código a seguir no final do arquivo.

```
#energia {  
    width: 140px;  
    height: 38px;  
    position: absolute;  
    left: 750px;  
    top: 7px;  
}
```

6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Com a formatação criada, a seguir vamos criar uma variável com o nome de energiaAtual que terá o valor inicial de três unidades, ou seja, a energia total do jogador.

7. No arquivo js.js, digite o código a seguir no local indicado pela Figura 91.

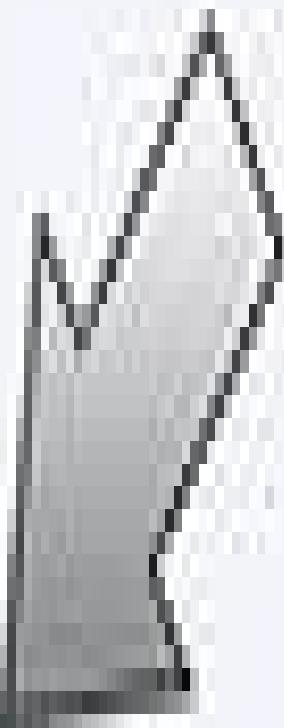
```
var energiaAtual=3;
```

var percent

var salvoes

var pedidos

var etiquetas



## **Figura 91**

Vamos iniciar subtraindo a variável energiaAtual toda vez que a div jogador entrar em colisão com a div inimigo1. Como você deve ter observado, iremos realizar alterações na função colisao.

8. Digite o código a seguir no local indicado pela Figura 92.

```
energiaAtual--;
```

```
if (collide.length > 0) {
    energyActual -= 1;
    initVelocityX = parseInt(S("initVelocityX"), CSS("left"));
    initVelocityY = parseInt(S("initVelocityY"), CSS("top"));
}
```

## **Figura 92**

Vamos realizar esse mesmo procedimento quando a div jogador entrar em colisão com a div inimigo2.

9. Digite o código a seguir no local indicado pela Figura 93.

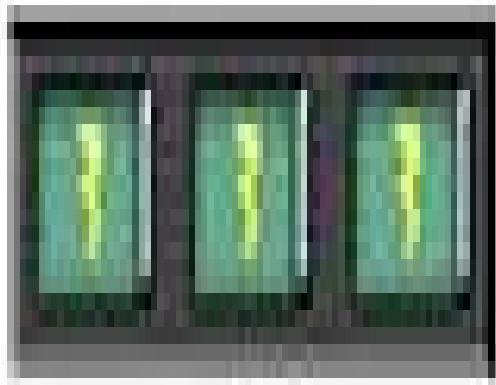
```
energiaAtual--;
```

```
if (coligao2.length>0) {  
    energiaactual = '  
    inimigo2X = parseInt($("#inimigo2").css("left"));  
    inimigo2Y = parseInt($("#inimigo2").css("top"));  
    explosao2(inimigo2X,inimigo2Y);  
}
```

### **Figura 93**

Para que não seja necessário atualizar o conteúdo da div energia toda vez que uma variável for atualizada, criaremos uma função que será chamada no game loop do jogo para deixar a div energia atualizada.

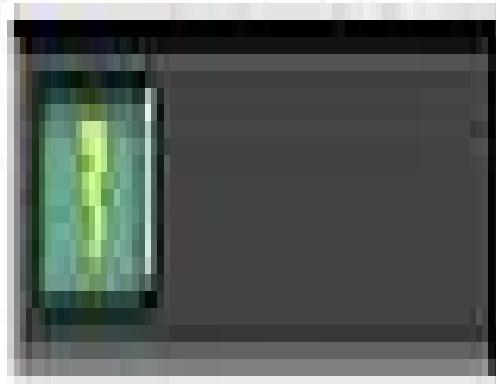
Nessa função verificaremos o valor da variável energiaAtual. Caso ela tenha o valor igual a 3, exibiremos no fundo da div energia a imagem energia3.png. Caso o valor da variável seja igual a 2, exibiremos a imagem energia2.png. Caso seja igual a 1, exibiremos a imagem energia1.png. Caso seja igual a 0, exibiremos a imagem energia0.png. Veja a Figura 94.



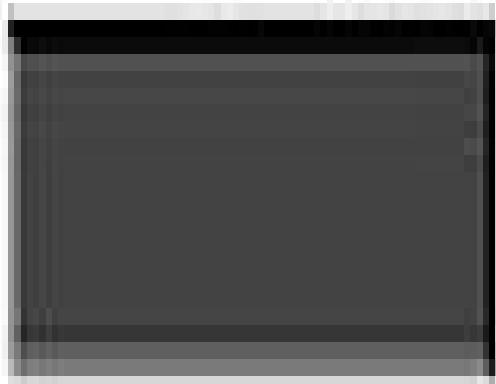
energia3.png



energia2.png



energia1.png



energia0.png

**Figura 94**

10. Digite o código a seguir no local indicado pela Figura 95.

energia();

```
function loop() {
```

```
    movefundo();
```

```
    movejogador();
```

```
    moveanimigo1();
```

```
    moveanimigo2();
```

```
    moveanigo();
```

```
    colisao();
```

```
    placar();
```

```
    energia();
```

```
} // Fim da função loop()
```

## **Figura 95**

11. Agora vamos criar a função energia. Digite o código a seguir no final do arquivo, mas dentro da função start, como indicado na Figura 96.

```
//Barra de energia
```

```
function energia() {
```

```
    if (energiaAtual==3) {
```

```
        $("#energia").css("background-image", "url(imgs/energia3.png)");
```

```
}
```

```
    if (energiaAtual==2) {
```

```
        $("#energia").css("background-image", "url(imgs/energia2.png)");
```

```
}
```

```
if (energiaAtual==1) {  
  
    $("#energia").css("background-image", "url(imgs/energia1.png)");  
  
}  
  
if (energiaAtual==0) {  
  
    $("#energia").css("background-image", "url(imgs/energia0.png)");  
  
//Game Over  
  
}  
  
} // Fim da função energia()
```

```
    } //fim da função placar()

    //Barra de energia
    function energia() {

        if (energiaAtual==3) {

            $("#energia").css("background-image", "url(imgs/energia3.png)");
        }

        if (energiaAtual==2) {

            $("#energia").css("background-image", "url(imgs/energia2.png)");
        }

        if (energiaAtual==1) {

            $("#energia").css("background-image", "url(imgs/energia1.png)");
        }

        if (energiaAtual==0) {

            $("#energia").css("background-image", "url(imgs/energia0.png)");

            //Game Over
        }

    } // Fim da função energia()

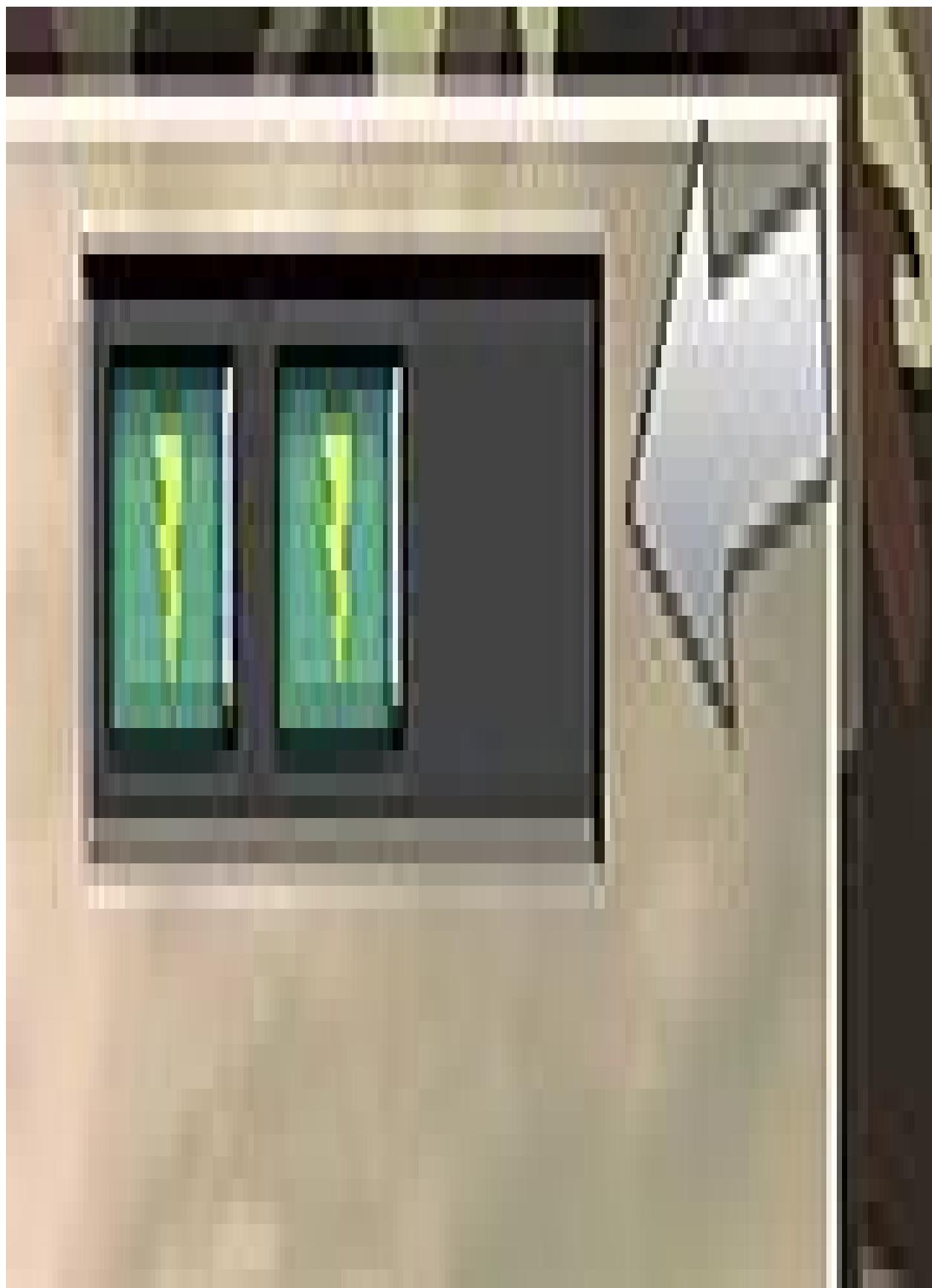
} // Fim da função start
```

## **Figura 96**

12. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

13. Execute o arquivo index.html pelo Google Chrome.

Observe que a imagem de fundo da div energia é atualizada toda vez que a div jogador entrar em colisão com a div inimigo1 ou a div inimigo2 (Figura 97), mas o jogo não é finalizado quando o jogador perde toda a sua energia. Criaremos a função de game over mais à frente.



**Figura 97**

## Capítulo 16

### Aumentando a dificuldade do jogo

O que vamos fazer nesse momento é dificultar um pouco a vida do jogador. A cada helicóptero que ele atingir (div inimigo1), iremos incrementar a dificuldade do jogo aumentando a velocidade do helicóptero.

Observe que, na função moveinimigo1, a velocidade da div inimigo1 no eixo X está vinculada ao valor da variável velocidade:

```
$("#inimigo1").css("left",posicaoX-velocidade);
```

Para aumentar a dificuldade do jogo iremos aumentar o valor da variável velocidade toda vez que houver colisão entre a div disparo com a div inimigo1.

1. Digite o código a seguir no local indicado na Figura 98.

```
velocidade=velocidade+0.3;
```

```
// Disparo com o inimigo1
```

```
if (colisao3.length>0) {  
    velocidade=velocidade+0.3;  
    pontos=pontos+100;
```

## **Figura 98**

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
3. Execute o arquivo index.html pelo Google Chrome.

Observe que agora toda vez que a div disparo entrar em colisão com a div inimigo1, a velocidade da animação da div inimigo1 será aumentada.



**Figura 99**

## Capítulo 17

### Som

Utilizando o elemento `<audio>` do HTML5 e os recursos do JavaScript para referenciar e executar o arquivo de áudio, vamos inserir uma música de fundo para o jogo e executar sons em determinadas colisões.

1. Abra o arquivo `index.html`.
2. Digite o código a seguir no local indicado na Figura 100.

```
<!-- Sons do jogo !-->

<audio src="sons/som.mp3" preload="auto" id="somDisparo"></audio>

<audio src="sons/explosao.mp3" preload="auto" id="somExplosao"></audio>

<audio src="sons/musica_fundo.mp3" preload="auto" id="musica"></audio>

<audio src="sons/gameover.mp3" preload="auto" id="somGameover"></audio>

<audio src="sons/perdido.mp3" preload="auto" id="somPerdido"></audio>

<audio src="sons/resgate.mp3" preload="auto" id="somResgate"></audio>
```

```
</div> <!-- Fim da div fundoGame -->  
</div> <!-- Fim da div container -->
```

```
<!-- Sons do jogo -->  
<audio src="sons/som.mp3" preload="auto" id="somSparo"></audio>  
<audio src="sons/explosao.mp3" preload="auto" id="somExplosao"></audio>  
<audio src="sons/musica_fundo.mp3" preload="auto" id="musica"></audio>  
<audio src="sons/gameover.mp3" preload="auto" id="somGameover"></audio>  
<audio src="sons/perdido.mp3" preload="auto" id="somPerdido"></audio>  
<audio src="sons/resgate.mp3" preload="auto" id="somResgate"></audio>
```

```
</body>
```

```
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
```

## **Figura 100**

3. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar no arquivo js.js as variáveis que controlarão a execução dos sons.

4. Abra o arquivo js.js e digite o código a seguir no local indicado na Figura 101.

```
var somDisparo=document.getElementById("somDisparo");
var somExplosao=document.getElementById("somExplosao");
var musica=document.getElementById("musica");
var somGameover=document.getElementById("somGameover");
var somPerdido=document.getElementById("somPerdido");
var somResgate=document.getElementById("somResgate");
```

```
var pontos=0;  
var salvos=0;  
var perdidos=0;  
var energiaAtual=3;
```

```
var somDisparo=document.getElementById("somDisparo");  
var somExplosao=document.getElementById("somExplosao");  
var musica=document.getElementById("musica");  
var somGameover=document.getElementById("somGameover");  
var somPerdido=document.getElementById("somPerdido");  
var somResgate=document.getElementById("somResgate");
```

## **Figura 101**

Inicialmente criaremos o código que vai executar a música de fundo. Essa música será executada em um loop, ou seja, assim que o arquivo de áudio chegar ao seu final, ele será reiniciado.

5. Digite o código a seguir logo abaixo do código digitado anteriormente.

```
//Música em loop
```

```
musica.addEventListener("ended", function(){ musica.currentTime = 0;  
musica.play(); }, false);
```

```
musica.play();
```

6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

7. Execute o arquivo index.html pelo Google Chrome.

Observe que, quando o jogo é iniciado, a música de fundo será executada. O próximo passo é executar o som correspondente a cada colisão.

8. Inicialmente vamos executar o som do disparo. Digite o código a seguir no local indicado na Figura 102.

```
somDisparo.play();
```

```
function dispazo() {
```

```
if (pedeActVar==true) {
```

```
pedeActVar=false;
```

```
sonDispazo.play();
```

## **Figura 102**

O próximo som a ser inserido é o som do resgate, que será executado quando a div jogador entrar em colisão com a div amigo.

9. Digite o código a seguir no local indicado na Figura 103.

```
somResgate.play();
```

```
// jogador com o amigo
```

```
if (colisoes.length>0) {
```

```
sonResgate.play();
```

```
salvo++;
```

```
reposicionaAmigo();
```

```
sf("amigo").remove();
```

```
}
```

### **Figura 103**

Os demais sons serão executados nas funções explosao1, explosao2 e explosao3. Vamos iniciar executando o som somExplosao na função explosao1.

10. Digite o código a seguir no local indicado na Figura 104.

```
somExplosao.play();
```

```
//Explode !
```

```
function explode1(inimigoX,inimigoY) {
    var audioExplode = new Audio("audio/explode.mp3");
    audioExplode.play();
    $('#fundocante').append("<div id='explode1'></div>");
    var divExplode = $('#explode1');
    divExplode.css("background-color", "red");
    divExplode.css("width", "20px");
    divExplode.css("height", "20px");
    divExplode.css("border-radius", "50%");
    divExplode.css("position", "absolute");
    divExplode.css("left", inimigoX);
    divExplode.css("top", inimigoY);
    divExplode.css("border", "1px solid black");
    divExplode.css("border-bottom-color", "white");
    divExplode.css("border-top-color", "white");
    divExplode.css("border-left-color", "white");
    divExplode.css("border-right-color", "white");
    divExplode.css("border-top-left-radius", "50% 0% 0% 0% / 50%");
    divExplode.css("border-top-right-radius", "0% 50% 0% 0% / 50%");
```

**Figura 104**

11. Repita esse procedimento com a função explosao2, como indicado na Figura 105.

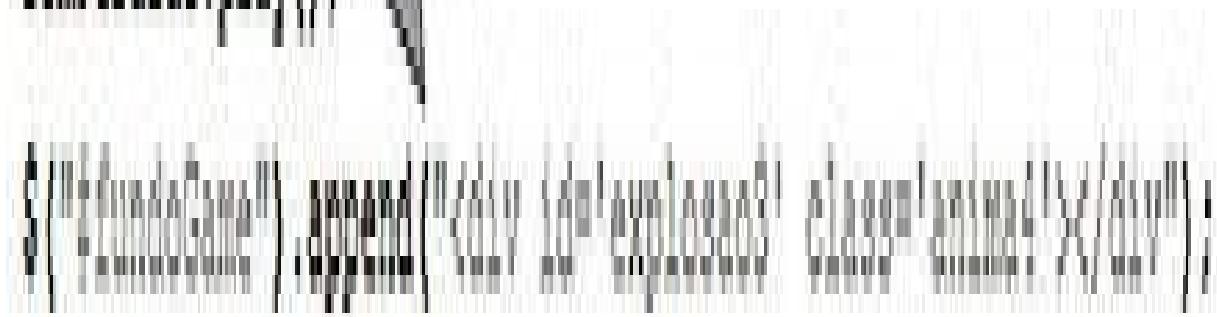
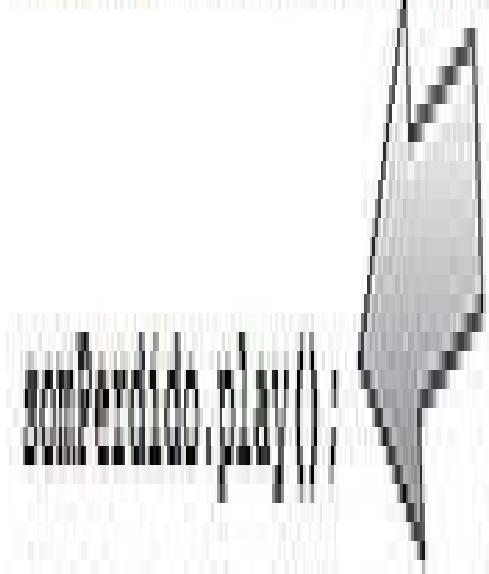
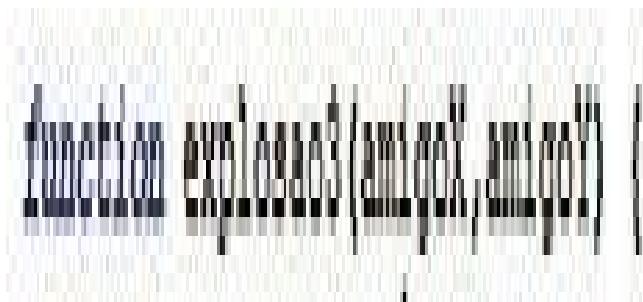
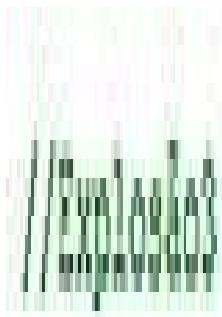
```
//Explosão2
```

```
function explosao2(inimigo2X,inimigo2Y) {
    sonExplosao.play();
    $($("#LunduGame"),append("<div id='explosao2'></div>");
```

### **Figura 105**

12. Na função explosao3 executaremos o som somPerdido. Digite o código a seguir no local indicado na Figura 106.

```
somPerdido.play();
```



## **Figura 106**

Vamos testar o funcionamento do código.

13. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
14. Execute o arquivo index.html pelo Google Chrome.

Observe que os sons serão executados.

Você teve ter notado que o som somGameover não foi indicado no código. Realizaremos esta operação no próximo capítulo, quando criarmos a função gameOver.

## Capítulo 18

### ***Game over***

Neste capítulo criaremos a função gameOver. Essa função inicialmente alterará o valor da variável fimdejogo para true, pausará a música de fundo, executará o som somGameover, removerá as divs jogador, inimigo1, inimigo2 e amigo e criará uma nova div onde será exibida a pontuação do jogador. Mas a principal operação da função gameOver será parar a função gameloop.

1. Abra o arquivo js.js.
2. Digite o código a seguir no local indicado na Figura 107.

```
//Função GAME OVER

function gameOver () {

fimdejogo=true;

musica.pause();

somGameover.play();

window.clearInterval(jogo.timer);

jogo.timer=null;
```

```
$( "#jogador" ).remove();
$( "#inimigo1" ).remove();
$( "#inimigo2" ).remove();
$( "#amigo" ).remove();

$( "#fundoGame" ).append("<div id='fim'></div>");

$( "#fim" ).html("<h1> Game Over </h1><p>Sua pontuação foi: " + pontos +
"</p>" + "<div id='reinicia' onClick=reiniciaJogo()><h3>Jogar
Novamente</h3></div>");

} // Fim da função gameOver();
```

```
) // Fim da função energia()
```

```
//Função GAME OVER
```

```
function gameOver () {
```

```
fimdejogo=true;
```

```
musica.pause();
```

```
soundGameover.play();
```

```
window.clearInterval(jogo.timer);
```

```
jogo.timer=null;
```

```
$("#jogador").remove();
```

```
$("#inimigo1").remove();
```

```
$("#inimigo2").remove();
```

```
$("#amigo").remove();
```

```
$("#fundoGame").append("<div id='fim'></div>");
```

```
$("#fim").html("<h1> Game Over </h1><p>Sua pontuação foi: " + pontos + "</p>" + "<div id='reinicia' onClick=reiniciaJogo()><h3>Jogar Novamente</h3></div>");
```

```
) // Fim da função gameOver();
```

```
} // Fim da função start
```

## **Figura 107**

Agora precisamos chamar a função gameOver assim que a energia do jogador chegar ao fim.

3. Digite o código a seguir no local indicado na Figura 108.

```
gameOver();
```

```
if (playerHealth == 0) {
    // Game Over
    gameOver();
}

// End da função executa()
```

## **Figura 108**

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Para finalizar, vamos criar a formatação CSS da div fim, que será criada na função gameOver, e para o seletor h3, também utilizado na função.

5. Abra o arquivo estilos.css e digite o código a seguir no final do arquivo.

```
h3 {  
    font-family: Titulo;  
    font-size: 20px;  
    color: #603A03;  
}
```

```
#fim {  
    width: 350px;  
    height: 200px;  
    background-color: #FFF;  
    margin-left: auto;
```

```
margin-right:auto;  
margin-top:100px;  
text-align:center;  
padding:10px;  
}  
  
}
```

Vamos testar o funcionamento do código.

6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
7. Execute o arquivo index.html pelo Google Chrome.

Quando a energia do jogador chegar ao fim, a div fim será exibida (Figura 109), mas o jogo ainda não pode ser reiniciado, pois a função reinicia chamada na div fim ainda não foi criada.

# GAME OVER

Sua pontuação foi 1250

[JOGAR NOVAMENTE](#)

**Figura 109**

## Capítulo 19

### Reiniciando o jogo

Neste capítulo criaremos uma função que reinicia o jogo após a mensagem de game over ser exibida. Observe que, pelo código a seguir (inserido no capítulo anterior), chamamos uma função com o nome de reiniciaJogo pela propriedade onClick. Essa função será chamada quando o jogador clicar sobre o texto “Jogar Novamente”.

```
$("#fim").html("<h1> Game Over </h1><p>Sua pontuação foi: " + pontos +  
"</p>" + "<div id='reinicia' onClick=reiniciaJogo()><h3>Jogar  
Novamente</h3></div>");
```

1. Abra o arquivo js.js.

O detalhe importante da função reiniciaJogo é que ela deve ser criada fora da função start.

2. Digite o código a seguir no local indicado na Figura 110.

```
//Reinicia o Jogo
```

```
function reiniciaJogo() {  
    somGameover.pause();  
    $("#fim").remove();  
    start();  
  
} //Fim da função reiniciaJogo
```

```
    } // Fim da função gameOver();  
  
}  
} // Fim da função start
```

//Reinicia o Jogo

```
function reiniciaJogo() {  
    somGameover.pause();  
    $('#fim').remove();  
    start();  
  
}  
} //Fim da função reiniciaJogo
```

**Figura 110**



Observe que, nesta função, inicialmente pausamos a música executada na função gameOver.

```
somGameover.pause();
```

A seguir a div fim é removida e a função start é executada, novamente reiniciando o jogo.

```
$("#fim").remove();
```

```
start();
```

Vamos testar o funcionamento do código.

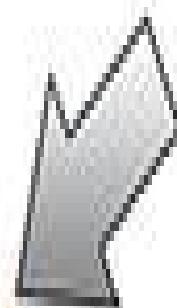
3. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

4. Execute o arquivo index.html pelo Google Chrome.

Observe que, ao clicar no texto “Jogar Novamente”, o jogo é reiniciado (Figura 111).

# GAME OVER

Sua pontuação foi 2100



## JOGAR NOVAMENTE

**Figura 111**

## Capítulo 20

# Criando uma versão do jogo para dispositivos móveis

Neste capítulo vamos criar uma versão do jogo para dispositivos móveis do tipo tablet.

Para criar essa versão, iremos adaptar os comandos de movimentação do helicóptero pelo toque (touch). Para que não seja necessário pressionar botões para realizar os disparos, faremos com que os disparos sejam realizados automaticamente, melhorando a jogabilidade.

## **Utilizando uma biblioteca para detectar toques**

Hammer é uma biblioteca JavaScript utilizada para identificar gestos multi-touch em dispositivos que suportam essa tecnologia, como tablets e celulares. Os gestos suportados pelo Hammer são: toque, toque duplo, arrastar e transformar (pinça para zoom) etc. A biblioteca Hammer pode ser baixada nesta URL:

<http://eightmedia.github.io/hammer.js/>

Não é necessário baixar o arquivo, pois ele já está disponível na pasta js do jogo.

1. Para que não seja necessário modificar o jogo já pronto, faça uma cópia de todo o conteúdo da pasta Resgate para uma pasta com o nome de Mobile.



## **Figura 112**

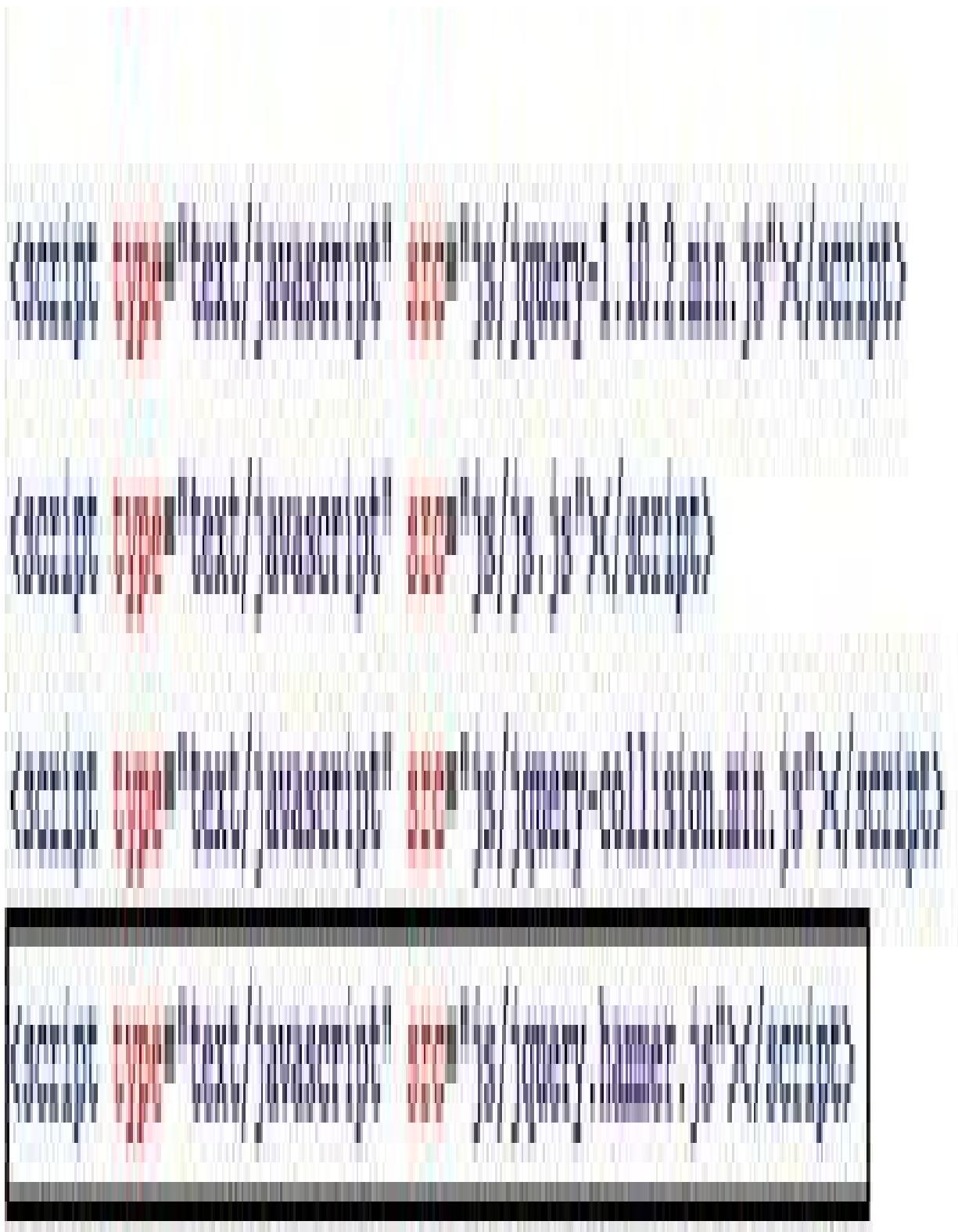
A partir de agora todas as alterações no código devem ser feitas nos arquivos da

■

O primeiro passo é vincular a biblioteca Hammer ao arquivo index.html.

2. Abra o arquivo index.html da pasta Mobile.
3. Digite o código a seguir no local indicado na Figura 113.

```
<script type="text/javascript" src="js/jquery.hammer.js"></script>
```



### **Figura 113**

4. Pressione as teclas Ctrl + S para salvar as alterações no código.

Agora vamos substituir a função que identifica as teclas pressionadas para uma função utilizando o código da biblioteca Hammer.

5. Abra o arquivo js.js da pasta Mobile.
6. Selecione o código indicado na Figura 114.

```
D: 68
```

```
}
```

```
jogo.pressionou = [];
```

```
//Verifica se o usuário pressionou alguma tecla
```

```
jogo.pressionou = [];
```

```
$(document) .keydown(function(e) {
```

```
    jogo.pressionou[e.which] = true;
```

```
});
```

```
$(document) .keyup(function(e) {
```

```
    jogo.pressionou[e.which] = false;
```

```
});
```

```
//Game Loop
```

```
jogo.timer = setInterval(loop,30);
```

## Figura 114

7. Pressione a tecla Delete para excluir o código.

8. Substitua pelo código indicado a seguir:

```
//Touch
```

```
$("#jogador")
```

```
.hammer({ drag_max_touches:0 })
```

```
.on("touch drag", function(ev) {
```

```
    var touches = ev.gesture.touches;
```

```
    ev.gesture.preventDefault();
```

```
    for(var t=0, len=touches.length; t<len; t++) {
```

```
        var target = $(touches[t].target);
```

```
        target.css({
```

```
            zIndex: 1337,
```

```
            top: touches[t].pageY-50
```

```
});
```

```
//Limita movimentação
```

```
var topo = parseInt($("#jogador").css("top"));
```

```
if (topo<=0) {
```

```
    $("#jogador").css("top",0);
```

```
}
```

```
if (topo>=410) {
```

```
    $("#jogador").css("top",434);
```

```
}
```

```
}
```

```
});
```

Esse é um código padrão de movimentação da biblioteca Hammer do tipo toca e arrasta (touch and drag). Esse código fará com que a div jogador seja movimentada pelo toque.

Caso você queira mais informações sobre o código e demais funções da biblioteca, acesse a documentação pela URL a seguir:

<https://github.com/EightMedia/hammer.js/wiki>

Observe que também indicamos o limite de movimentação da div jogador pela variável topo.

```
if (topo<=0) {  
    $("#jogador").css("top",0);
```

```
}
```

```
if (topo>=410) {  
    $("#jogador").css("top",434);
```

```
}
```

O próximo passo é chamar a função disparo no game loop do jogo, pois o

disparo não será mais realizado pela tecla D, e sim automaticamente. Também removeremos do game loop do jogo a função movejogador, pois a div jogador não será mais movimentada por essa função.

9. Substitua a função movejogador da função loop para disparo, como indicado na Figura 115.

```
function loop() {  
    movefundo();  
    disparo();  
    moveinimigo1();  
    moveinimigo2();  
    moveamigo();  
    colisao();  
    placar();  
    energia();  
}  
// Fim da função loop()
```

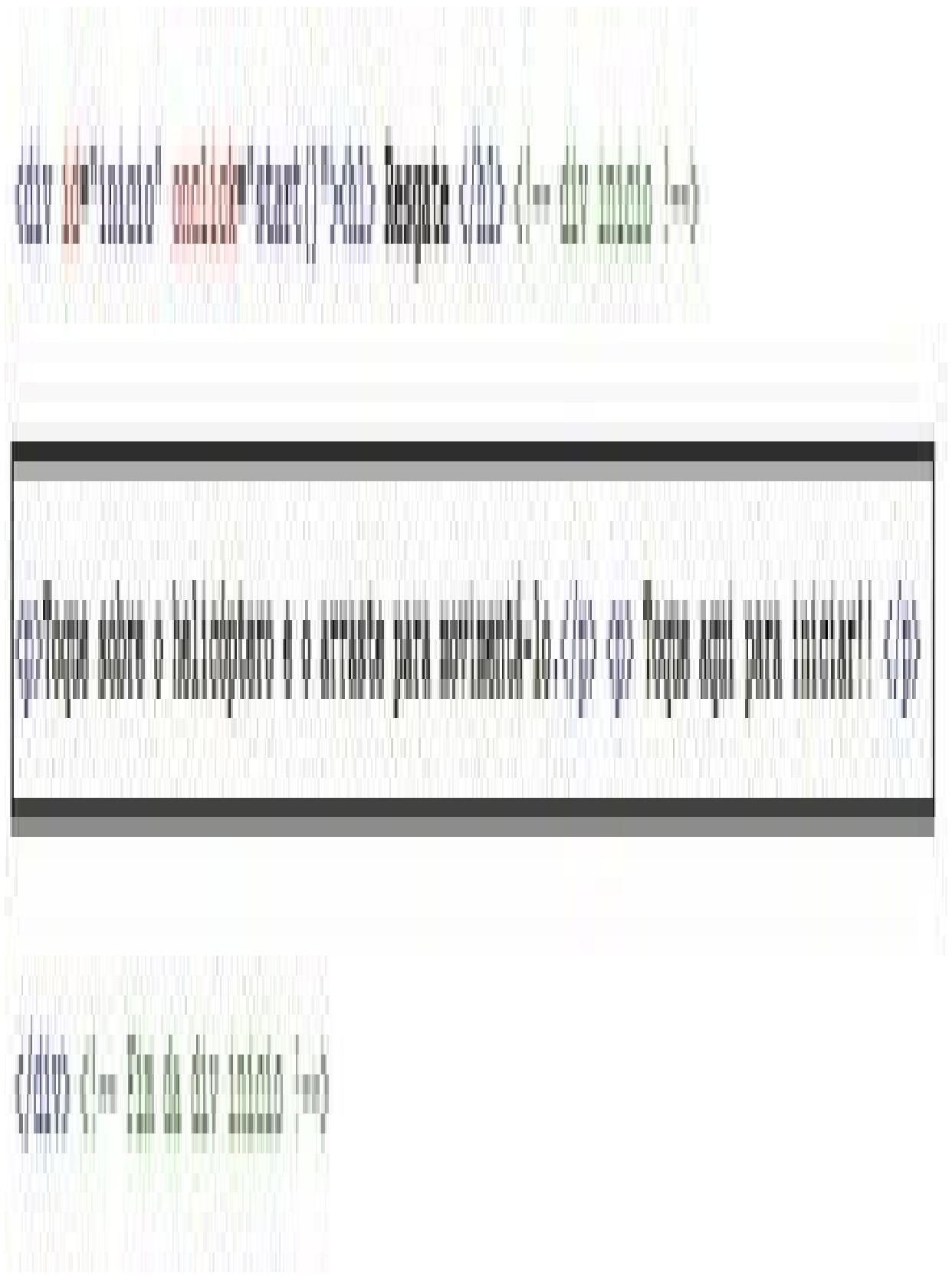
### **Figura 115**

A seguir, vamos substituir a mensagem inicial da div inicio, pois não iremos mais utilizar as teclas W, S e D para controlar o helicóptero.

10. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

11. Abra o arquivo index.html e substitua o texto da div inicio para o texto indicado a seguir. Em caso de dúvida, consulte a Figura 116.

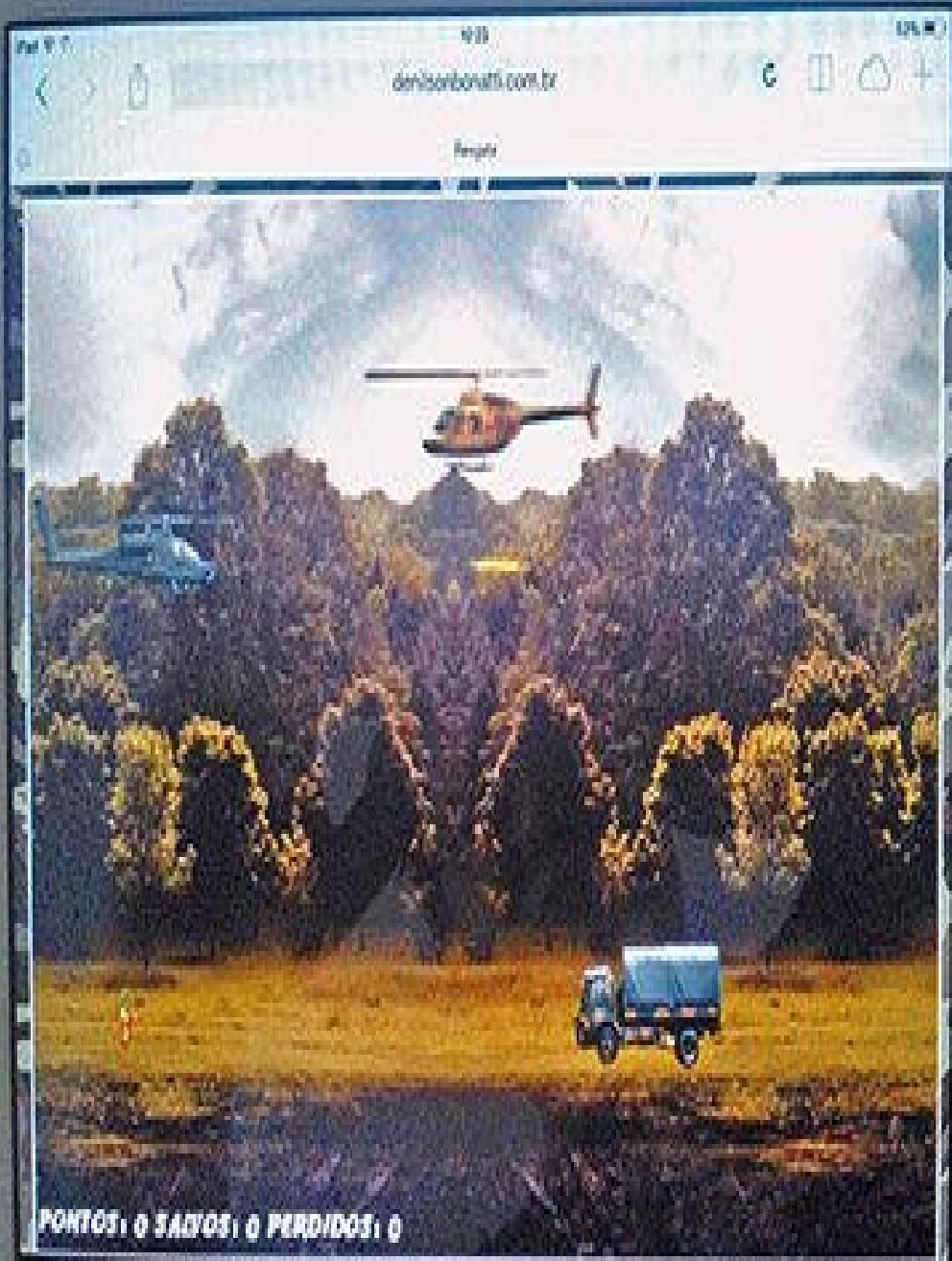
<p>Toque sobre o helicóptero e o arraste para movimentá-lo.</p> <p> Toque aqui para iniciar!! </p>



### **Figura 116**

12. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Na Figura 117 temos o jogo da pasta Mobile sendo executado em um iPad. Neste exemplo, os arquivos da pasta Mobile foram hospedados em um servidor web e acessados pelo browser.



### **Figura 117**

13. Execute o arquivo index.html pelo Google Chrome.

Veja que agora a movimentação do helicóptero pode ser realizada pelo toque e os disparos são realizados automaticamente.

-

Neste capítulo criamos uma versão do jogo para ser executada em um dispositivo

## **Capítulo 21**

## Facebook game

Com cerca de um bilhão de usuários ativos no mundo e quase oitenta milhões de usuários somente no Brasil, o Facebook é um grande potencial para os desenvolvedores de jogos.

Vantagens no desenvolvimento de jogos no Facebook:

- O desenvolvimento e a distribuição do jogo são de graça.
- Através do jogo é possível coletar dados de seus usuários, como data de nascimento, likes, fotos, entre outras informações relevantes.
- As atualizações realizadas podem ser publicadas no mural do usuário, aumentando assim a visibilidade do jogo.
- O Facebook oferece toda a documentação necessária para o desenvolvimento e a conexão com o seu ambiente gráfico, como likes (curtir), compartilhamentos etc.
- Após o desenvolvimento do jogo, o Facebook oferece ferramentas de publicidade para a distribuição do aplicativo.

Para desenvolver os jogos, o Facebook oferece uma grande quantidade de APIs. Uma API (Application Program Interface) fornece os caminhos para acessar certas funcionalidades do Facebook, como os botões “Curtir”, “Compartilhar”, entre outros. As APIs do Facebook também fornecem os meios de acesso às informações dos usuários, como conteúdo do seu mural, itens compartilhados e curtidos, entre outras funcionalidades. As APIs no Facebook estão divididas em duas categorias.

**Web Site API** – Fornece os meios de comunicação e integração com o Facebook através de um site. É com essa API que é possível inserir o botão “Curtir” em conteúdos externos e integrá-los ao Facebook.

**Mobile API** – Fornece os recursos necessários para integrar a comunicação de um aplicativo de dispositivo móvel com o Facebook.

As APIs do Facebook estão disponíveis em diversos kits de desenvolvimento (SDK). Você pode encontrar as APIs do Facebook para o SDK JavaScript, SDK PHP, SDK Android, SDK iOS, entre outros.

Criaremos uma versão do jogo utilizando o JavaScript SDK. Vamos realizar todo o procedimento, desde a conexão com o Facebook até a sua publicação no Facebook AppCenter, como indicado na Figura 118.



**Figura 118**

O jogo desenvolvido para Facebook pode ser acessado pelo link a seguir: [https:/](https://)

## **Criando uma conta de desenvolvedor no Facebook**

Para criar um jogo ou aplicativo para Facebook, antes é necessário se cadastrar como desenvolvedor, onde será possível administrar os jogos e aplicativos em desenvolvimento.

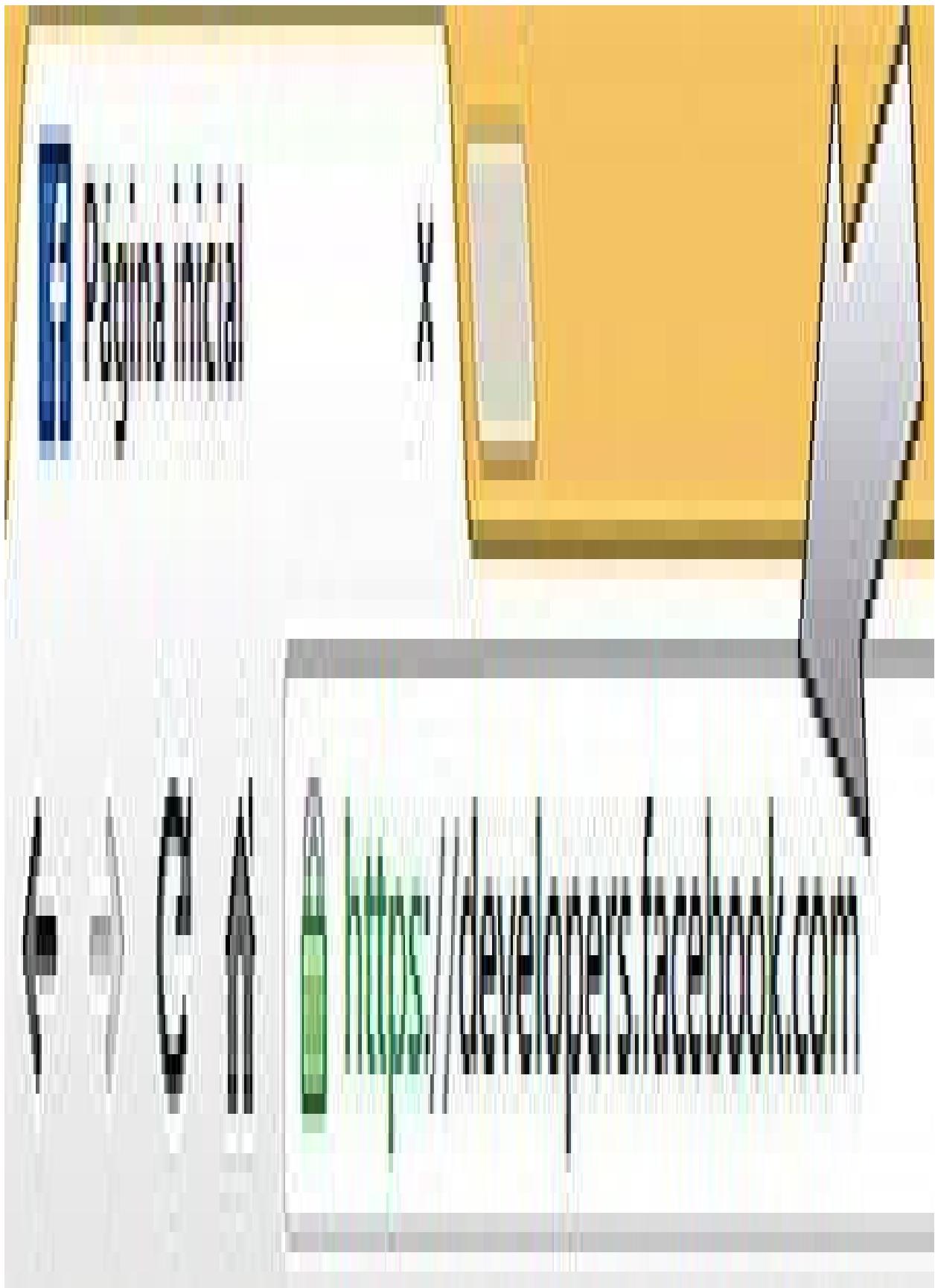
1. Inicie o Google Chrome e acesse a página do Facebook ([www.facebook.com.br](http://www.facebook.com.br)), como indicado na Figura 119.



**Figura 119**

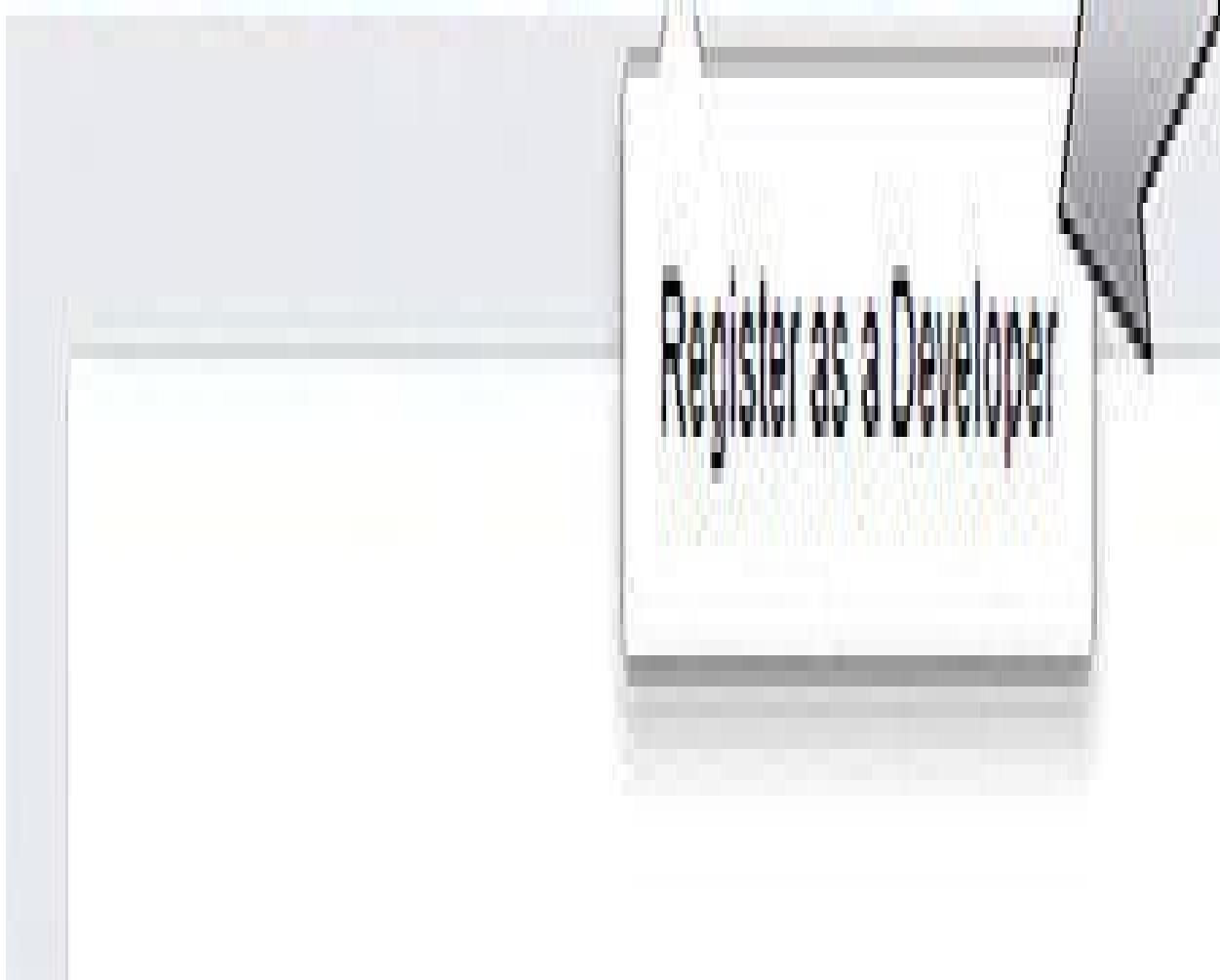
2. Acesse a sua conta de usuário do Facebook utilizando o seu nome de usuário e senha.

3. Após estar logado em sua conta no Facebook, acesse a URL [developers.facebook.com](https://developers.facebook.com), como indicado na Figura 120.



### **Figura 120**

4. Na guia “Aplicativos” selecione a opção “Register as a Developer”, como indicado na Figura 121.



### **Figura 121**

5. Será exibida uma nova janela, onde será solicitada a sua senha de acesso ao Facebook. Digite-a e clique no botão “Enviar”.

Digite sua senha novamente

X



Denilson Bonatti

Para a sua segurança, você deve inserir novamente sua senha para continuar.

Senha:

A rectangular input field for a password, with a grey border and a light grey background.

Enviar

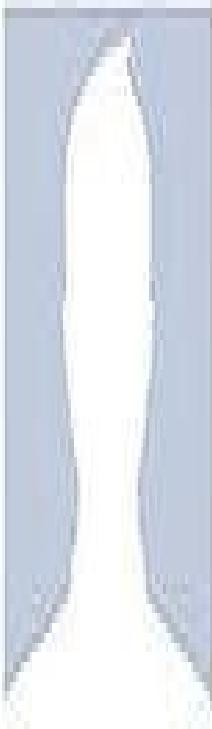
Cancelar

### **Figura 122**

6. Será exibida uma nova janela com a política de uso da área de desenvolvedor do Facebook. Altere a opção para “Sim”, como indicado na Figura 123, para aceitar os termos de acesso e clique no botão “Próxima”.

# Register as a Facebook Developer

X



Denilson Bonatti

Do you accept the Facebook Platform Policy  
and the Política de privacidade do Facebook



Sim

[Cancelar](#) [Próxima](#)

**Figura 123**

7. Cadastre um telefone celular para onde uma mensagem de texto com o código de ativação da conta será enviado (Figura 124).

# Register as a Facebook Developer

X

We need your Telefone to verify your account. It will be adicionado à  
linha do tempo, but won't be visible to your friends.

País

Telefone



Get Confirmation Code

Send as Text

Send via Phone Call

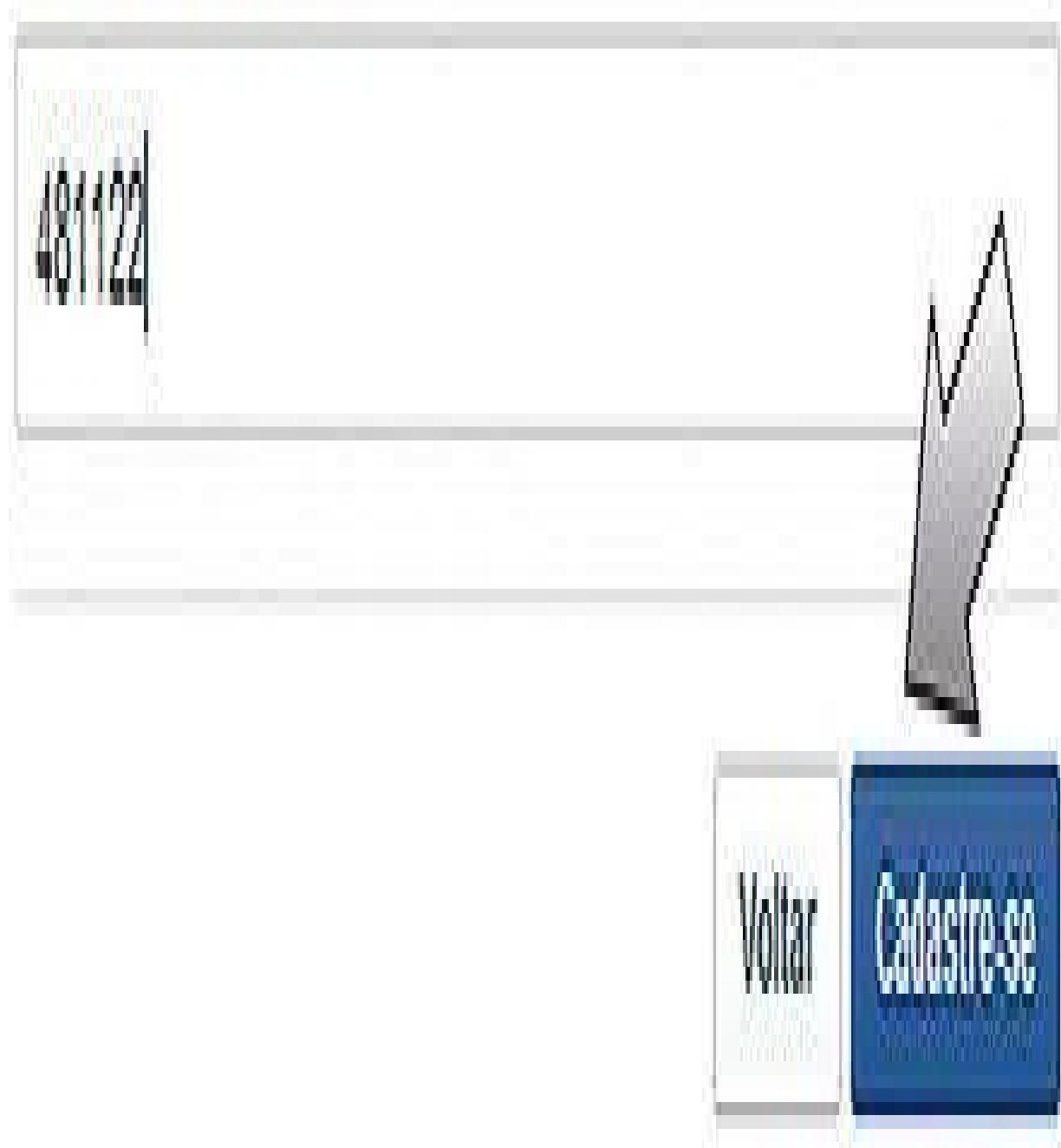
**Figura 124**

8. Clique no botão “Send as Text”.

Aguarde o código de ativação ser enviado para o telefone celular cadastrado.

9. Digite o código de ativação recebido e clique no botão “Cadastre-se” (Figura 125).

# Código de confirmación



**Figura 125**

10. Uma janela de boas-vindas será exibida. Clique no botão “Concluir”.

## **Criando um novo aplicativo**

O próximo passo é criar o um novo aplicativo no Facebook utilizando a sua conta de administrador.

1. Na guia Aplicativos selecione a opção “Create a New App”, como indicado na Figura 126.



## **Figura 126**

2. O primeiro passo é indicar o nome do aplicativo – neste caso, o nome do jogo que será criado. Dê um clique na caixa do item “Display Name” e digite o nome Resgate.

Na opção “Namespace” deve-se indicar o nome que será utilizado para identificação do jogo e a URL. Esse nome não pode conter espaços em branco, caracteres numéricos e especiais. É importante saber que o nome escolhido tem que estar disponível para uso, ou seja, não ter sido utilizado ainda no Facebook.

3. Observe na imagem a seguir que o nome Resgate foi utilizado.

# Create a New App

Get started integrating Facebook into your app or website

Display Name

Resgate

App Domains

resgate

### **Figura 127**

Com essa configuração o endereço URL do jogo seria:

[https://apps.facebook.com/resgate.](https://apps.facebook.com/resgate)

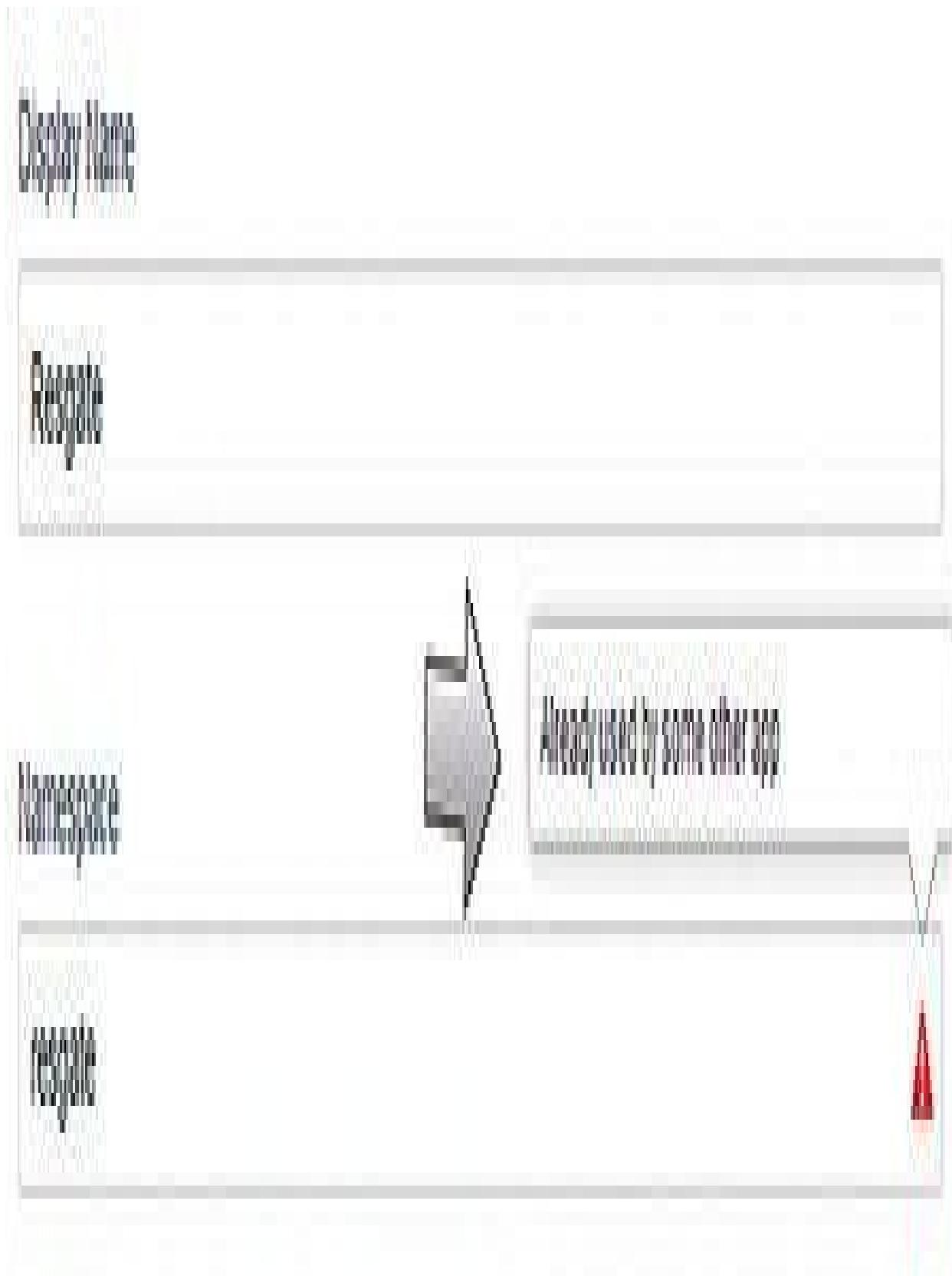
4. O próximo passo é selecionar a categoria do aplicativo. Dê um clique na guia de seleção do item “Categoria”.



## **Figura 128**

5. Selecione a categoria “Diversão”.
6. Clique no botão “Criar aplicativo”.

Caso o Namespace para o aplicativo não esteja disponível, uma mensagem de erro será exibida (Figura 129). Altere o nome na opção Namespace e clique no botão “Criar aplicativo”.



**Figura 129**

7. Digite o código de verificação e clique no botão “Enviar” (Figura 130).

## Verificação de segurança



Não é possível ler o texto acima?

Tente outro texto ou um captcha sonoro

Texto da caixa:

O que é isso?

Se você acha que está vendo isso por engano, avise-nos.

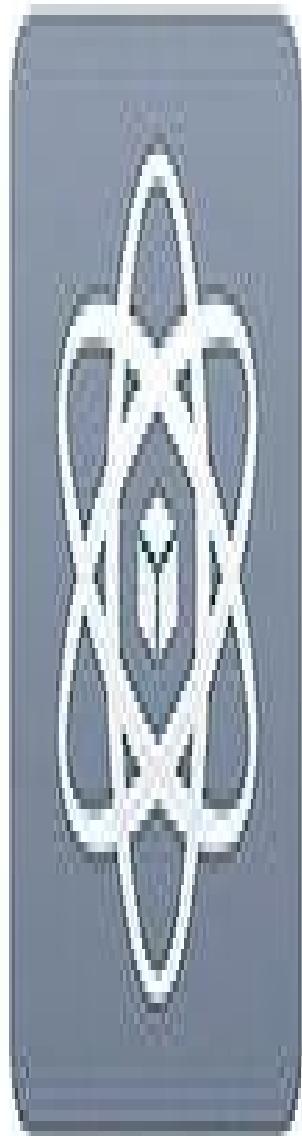
**Enviar**

**Cancelar**

### **Figura 130**

Observe que serão exibidos o nome do novo aplicativo e a sua identificação (ID). Essa identificação será muito importante para o desenvolvimento do aplicativo.

# Painel de controle



## Resgate

This app is in development mode (?)

ID do aplicativo

212195395653427

App Secret

XXXXXXXX

### **Figura 131**

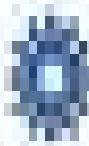
O próximo passo é indicar as configurações iniciais do aplicativo.

8. Dê um clique na opção “Configurações”, como indicado na Figura 132.

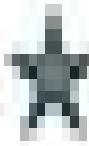
# Resgate



**Painel de controle**



**Configurações**



**Status & Review**

## **Figura 132**

A primeira configuração é indicar a URL do servidor local onde os arquivos estarão hospedados. Mas, antes disso, vamos conhecer alguns conceitos sobre o desenvolvimento de aplicativos para Facebook.

## **O que é um aplicativo para Facebook?**

Aplicativos para Facebook são acessados diretamente na página do Facebook. O game Angry Birds, por exemplo, é um dos milhares de aplicativos que podem ser acessados diretamente de uma conta no Facebook.

facebook

Request privacy, login & more



ANGRY BIRDS  
FRIENDS

NEW TOURNAMENT  
EVERY WEEK!

PLAY IT NOW!

ANGRY BIRDS  
STAR WARS

BETA

WEEKLY  
TOURNAMENT

Beat Tatooine-9  
TATOOINE

Angry Birds  
Friends

TATOOINE

DEATH STAR

SHOP

Trophies



### **Figura 133**

O que torna um aplicativo único é a integração com os dados do Facebook, como, por exemplo, a possibilidade de compartilhar a pontuação em um jogo com todos os amigos, ou, em aplicativo comercial, compartilhar novidades como novos produtos e serviços.

Ao adicionar um aplicativo no Facebook, ele será exibido no painel Aplicativos, na página inicial do usuário, onde é possível visualizar novas notificações do aplicativo.

## APLICATIVOS



Games

3



Hel War



Rescue your friends



Invasão do Planeta X



MeuFlashApp



Angry Birds Star Wars



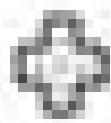
Observatório da Seleção



Nostalgia



DengueVille



Feed de jogos

20+

## **Figura 134**

**Todo aplicativo criado no Facebook recebe uma uma página específica.**

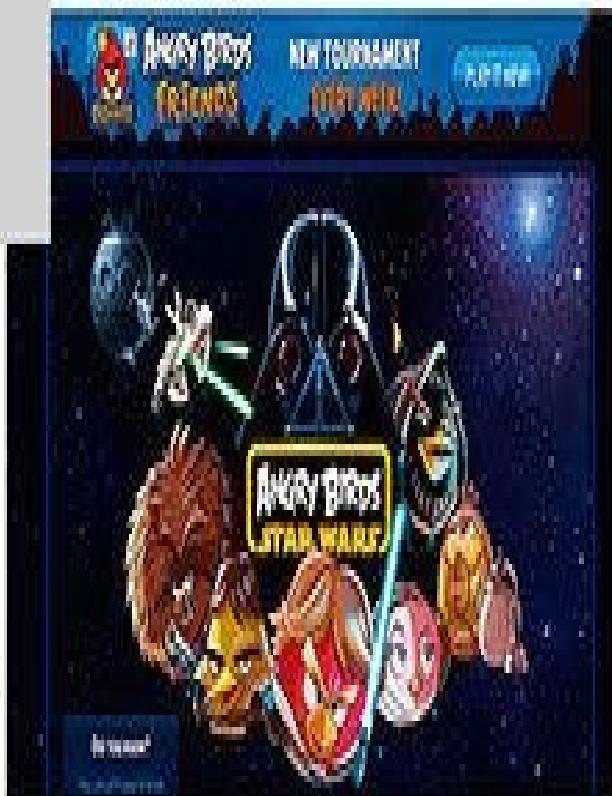
<https://apps.facebook.com/angrybirdsstarwars/>

O aplicativo é executado dentro da página em um local chamado de Canvas. O Canvas é um iFrame cujo conteúdo é executado em outro servidor. Os arquivos de um aplicativo para Facebook não são hospedados no servidor do Facebook. Para criar um aplicativo é necessário hospedá-lo em um servidor remoto. Dessa forma, o proprietário possui mais controle sobre o aplicativo. O Facebook apenas irá permitir o acesso às conexões sociais e informações do usuário do aplicativo.

CANVAS



Servidor Web



Arquivos  
HTML/JS/CSS

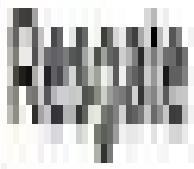
**Figura 135**

## Utilizando um servidor local

Para que não seja necessário hospedar os arquivos do jogo em um servidor remoto em seu desenvolvimento, executaremos os testes do jogo em um servidor local. Um servidor local, conhecido como localhost, é um servidor web instalado no computador. Para isso utilizaremos o aplicativo XAMPP, instalado anteriormente. Vamos configurar nosso primeiro aplicativo indicando como local de execução dos arquivos um servidor local.

1. Deixe o item “App Domains” inicialmente em branco para que possamos utilizar o servidor local, como indicado na Figura 136.

Dropbox



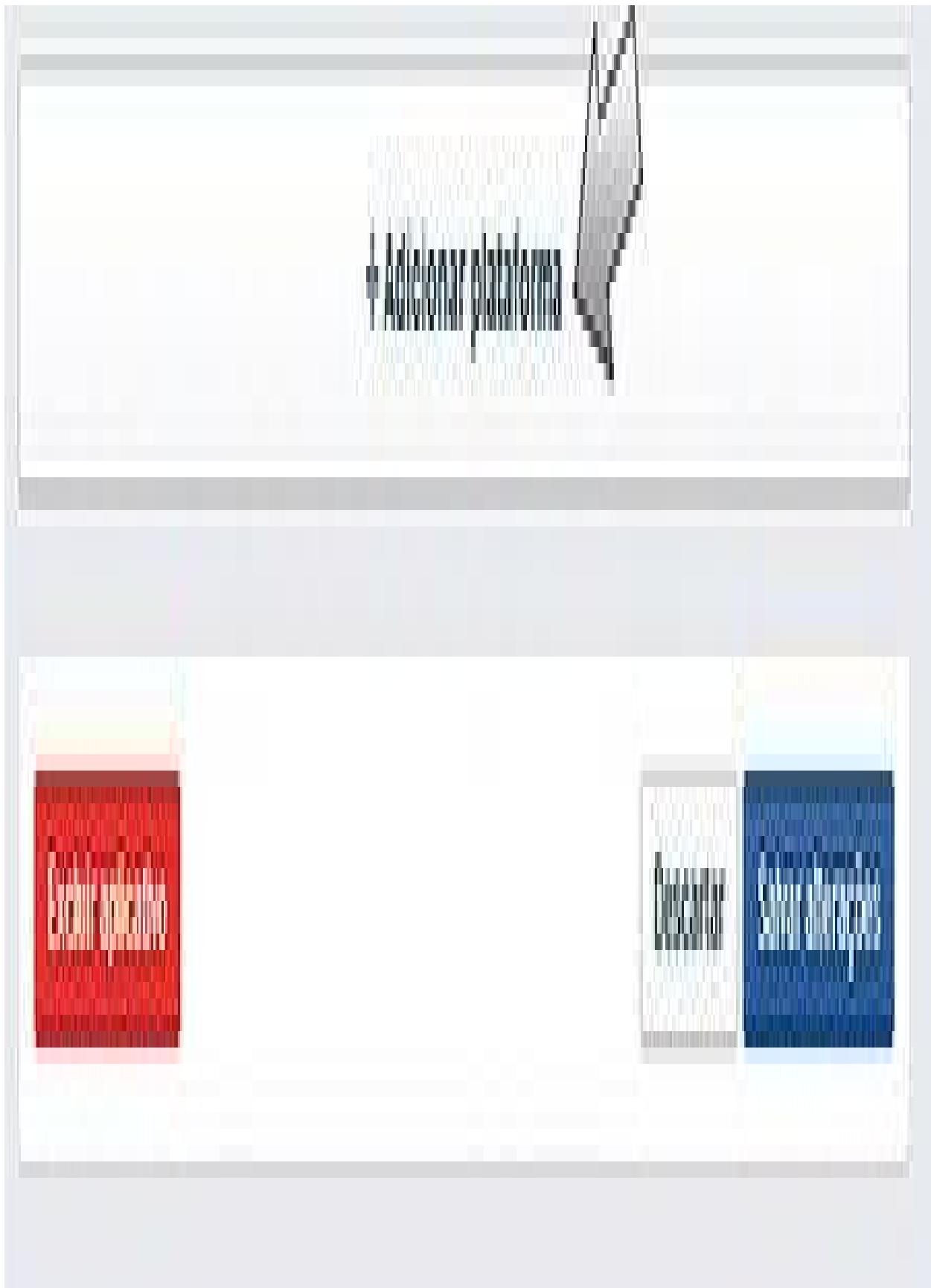
App Drawer



## **Figura 136**

O próximo passo é indicar o tipo de aplicativo que será criado. Nesse exemplo, criaremos um aplicativo que será executado diretamente na página do Facebook, chamado de Facebook App.

2. Dê um clique no item “Adicionar plataforma” (Figura 137).



**Figura 137**

3. Selecione a opção “App on Facebook”.

## Seleccionar plataforma



App on Facebook



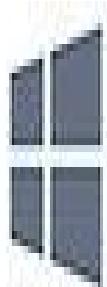
Site



iOS



Android



Windows App



Guia da página



Xbox



PlayStation

### **Figura 138**

4. Na opção “URL”, indique novamente o endereço do servidor do aplicativo: <http://localhost/facebook/>, como indicado na Figura 139.

## App on Facebook

### Página Canvas

<https://apps.facebook.com/resgateb>



Unity Integration

"Yes" if you have a Unity integration

URL

<http://localhost/facebook/>

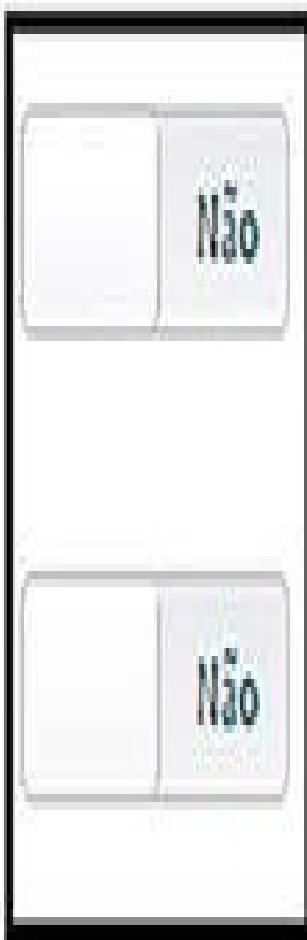


### **Figura 139**

A seguir vamos definir o tamanho do aplicativo. Para que ele se adeque ao Canvas do Facebook, selecione o modo fluido para a altura (height) e para largura (width) indicando “Não” nas opções descritas na Figura 140.

URL

<http://localhost/facebook/>



Canvas Fixed Width

"Yes" sets canvas width to 760 px.

Canvas Fixed Height

"Yes" allows setting fixed height.

**Figura 140**

5. Dê um clique no botão “Salvar alterações” (Figura 141).



## **Figura 141**

Essas são as configurações iniciais para o desenvolvimento do aplicativo. Futuramente vamos retornar às configurações para hospedá-lo em um servidor web.

O próximo passo é realizar uma cópia da pasta Resgate para a pasta de instalação do aplicativo XAMPP.

6. Dê um clique na pasta Resgate para selecioná-la (Figura 142).



### **Figura 142**

7. Copie esta pasta para a pasta de instalação do XAMPP (C:\xampp), dentro da pasta htdocs (Figura 143).

The screenshot shows a Windows File Explorer window with the following directory path:

```
Meu computador > OS (C:) > xampp > htdocs > Resgate
```

The 'Resgate' folder is selected, indicated by a blue selection bar. A large, semi-transparent gray arrow points from the top right towards the 'Resgate' folder. The file list table has columns for 'Nome' (Name) and 'Data de modifi' (Modification date).

Nome	Data de modifi
forbidden	03/02/2014 15:2
img	03/02/2014 15:2
Resgate	03/02/2014 15:2
restricted	03/02/2014 15:2
xampp	03/02/2014 15:2

**Figura 143**

8. Renomeie a pasta Resgate para facebook. Utilize todos os caracteres em minúsculo, como indicado na Figura 144.

Nome



**Figura 144**

## Executando o jogo no servidor local

Vamos executar o jogo no servidor local utilizando o XAMPP.

1. Dê um clique duplo no arquivo xampp-control encontrado na pasta C:\xampp (Figura 145).



uninstall



xampp\_shell



xampp\_start



xampp\_stop



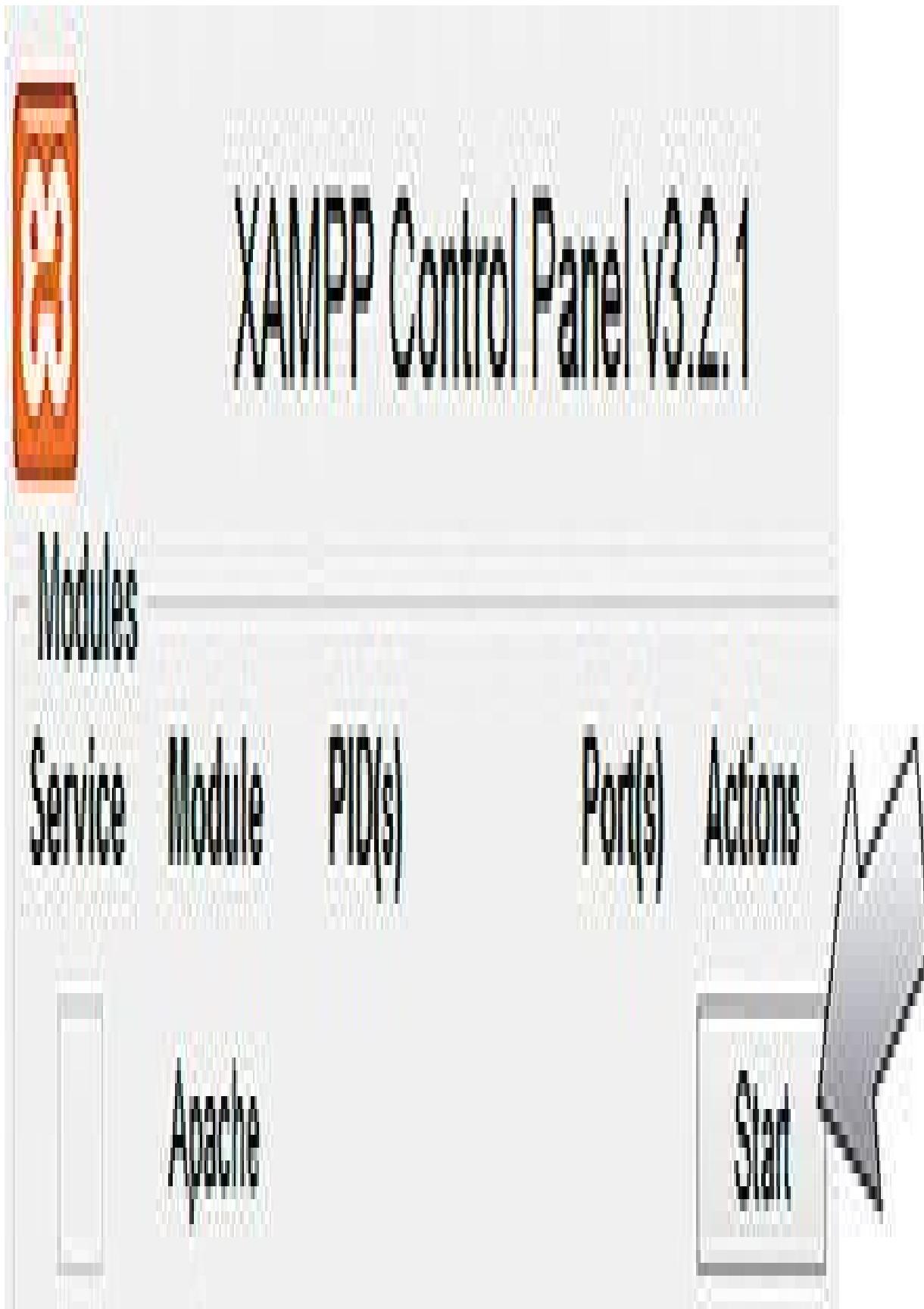
xampp-control



xampp-control

**Figura 145**

2. Dê um clique no botão “Start” do módulo Apache, indicado na Figura 146.



## **Figura 146**

Caso você esteja utilizando o Skype, feche-o e execute o módulo Apache. Depo

■ Observe que o módulo Apache está em execução (Figura 147).



# KAMPP Control Panel V3.2.1

Modules

Service	Module	PID(s)	Port(s)	Actions
	Apache	1560 2288	80, 443	Stop

### **Figura 147**

Agora vamos testar a execução do jogo pelo servidor local.

3. Abra o Google Chrome e digite a URL indicada a seguir:

`http://localhost/facebook/`

Observe que o jogo da pasta facebook será executado (Figura 148).



### **Figura 148**

Com o jogo sendo executado no servidor local, podemos dar início ao código que realizará a comunicação do jogo com o Facebook.

## Capítulo 22

### Criando uma conexão com o Facebook

Para utilizarmos os recursos do Facebook, como identificar o nome do jogador, compartilhar pontuação no mural etc., é necessário realizar a conexão do jogo com o Facebook. Para tal, utilizaremos o Facebook JavaScript SDK, que é um kit de desenvolvimento fornecido pelo Facebook para realizar a integração do aplicativo com os dados dos usuários e o ambiente gráfico do Facebook.

O SDK fornece uma série de funções:

- Gerenciamento do botão “Curtir” e outros plugins sociais.
- Gerenciamento de logon e logoff do aplicativo.
- Gerenciamento das caixas de diálogo para compartilhamento de informações etc.
- Busca entre amigos do usuário.

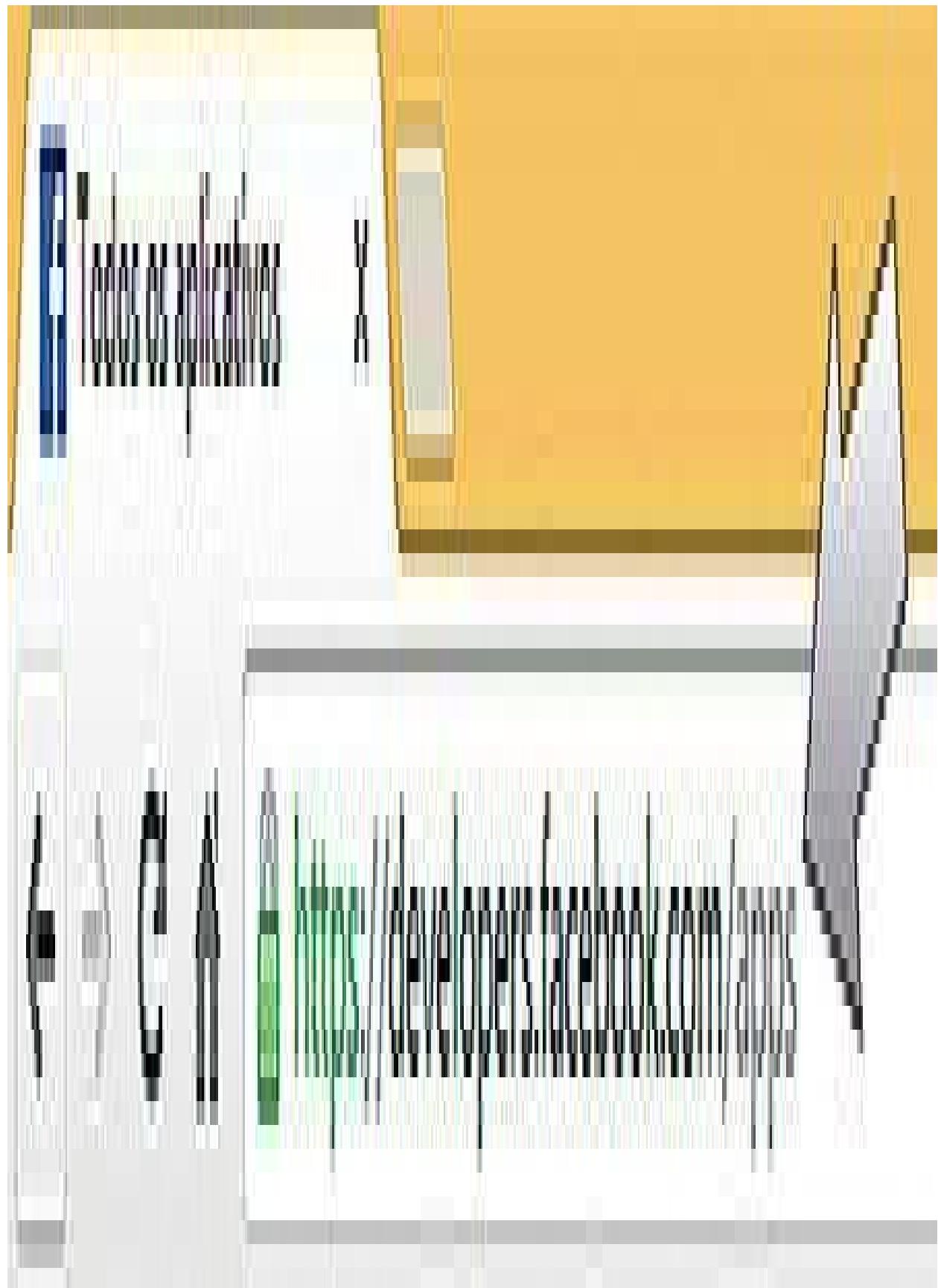
# Configurando JavaScript SDK

Vamos configurar o Facebook JavaScript SDK. Essa configuração será realizada no arquivo js.js.

Antes de editar o arquivo js.js vamos identificar o número do ID de aplicativo. Esse número é necessário para realizar a conexão do aplicativo com o Facebook.

1. Abra o Google Chrome e digite a URL a seguir.

<https://developers.facebook.com/apps>



### **Figura 149**

2. Dê um clique sobre o aplicativo Resgate (Figura 150).

# Resgate

Esse aplicativo está em modo resgate



0

Novos usuários

0% de cotação

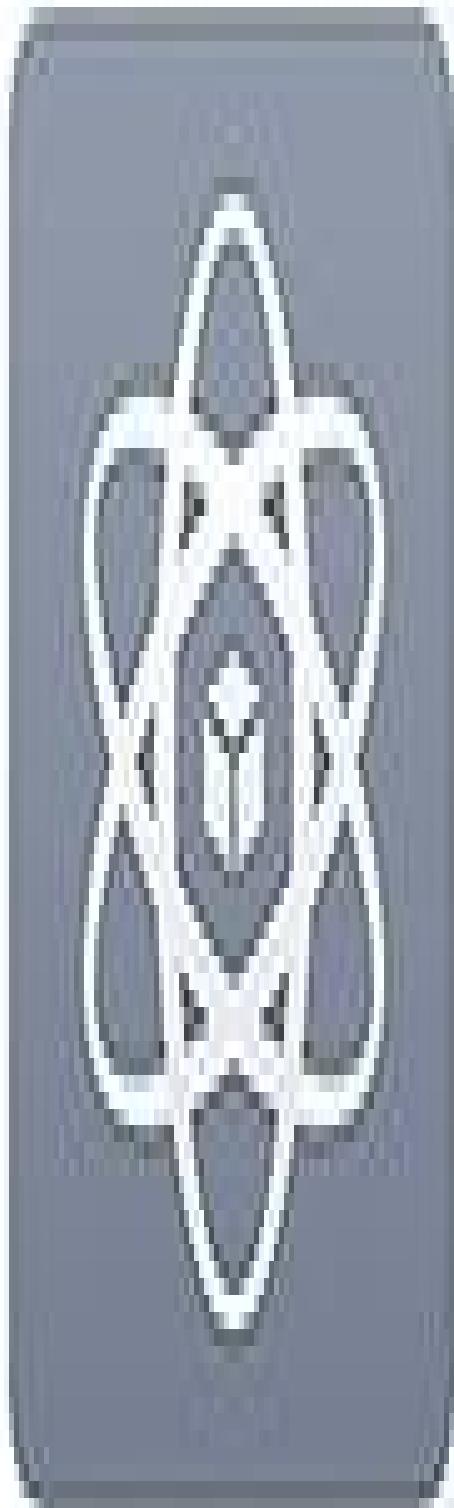
2

Usuários ativos mensalmente

0% de cotação

**Figura 150**

3. Anote o número indicado no item “ID do aplicativo” (Figura 151).

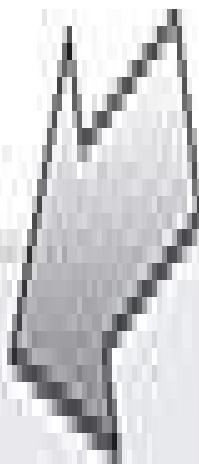


# Resgate

This app is in development mode

0 do aplicativo

479967890466753



## **Figura 151**

No arquivo js.js vamos criar uma função que irá carregar e iniciar o JavaScript SDK.

A conexão do seu aplicativo com o JavaScript SDK deverá ser realizada com o método FB.init. Essa função realizará a conexão do aplicativo com o JavaScript SDK através do ID do aplicativo.

O código de conexão apresentado a seguir é fornecido pela documentação do Facebook e pode ser acessado pelo link:

<https://developers.facebook.com/docs/javascript/quickstart/v2.0>

4. Abra o arquivo js.js da pasta facebook.

5. Digite o código JavaScript a seguir no início do arquivo, como indicado na Figura 152.

//Conexão com o Facebook

```
window.fbAsyncInit = function() {  
  FB.init({  
    appId : '429967890466753', // App ID
```

```
  xfbml: true,
```

```
  version: 'v2.0'
```

```
});
```

```
//Definições de logon
```

```
};
```

```
//Conexão com o Facebook
window.fbAsyncInit = function() {
    FB.init({
        appId: '429967890466753', //App ID
        xfbml: true,
        version: 'v2.0'
    });

    //Definições de logon
};

function start() { // Início da função start()
    $("#inicio").hide();
    $("#fundosame").append("<div id='jogador' class='animal'></div>");
}
```

**Figura 152**



Os seguintes parâmetros foram indicados na construção do método FB.init:

appId: '429967890466753'

Neste parâmetro deve ser indicado o número do ID do aplicativo. Altere essa numeração para o ID apresentado na definição do novo aplicativo.

version: 'v2.0'

Indica a versão do Facebook API que será utilizada no código. Neste exemplo estamos usando a versão 2.0.

frictionlessRequests: true

Caso a propriedade esteja definida como true, as requisições para outros usuários logarem nesse aplicativo poderão ser realizadas. Essas requisições poderão ser realizadas pelo método FB.ui, que aprenderemos mais à frente.

xfbml: true

O parâmetro xfbml definido como true habilita a comunicação com o XFBML (eXtended Markup Language FaceBook). O XFBML é uma linguagem de apresentação de conteúdo que possibilita aos programadores web integrarem

funcionalidades do Facebook em páginas web. Para que seja possível visualizar alguns plugins sociais do Facebook, como botão “Curtir” e caixa de comentários, esse parâmetro deve estar habilitado como true.

Com as definições de logon determinadas, o próximo passo é criar o código de conexão com JavaScript SDK.

6. Digite o código a seguir logo após o código digitado anteriormente.

```
// Carrega o JavaScript SDK

(function(d, s, id){

var js, fjs = d.getElementsByTagName(s)[0];
if (d.getElementById(id)) {return;}
js = d.createElement(s); js.id = id;
js.src = “//connect.facebook.net/pt_BR/sdk.js”;
fjs.parentNode.insertBefore(js, fjs);
})(document, ‘script’, ‘facebook-jssdk’));
```

```
//Conexão com o Facebook
) window.fbAsyncInit = function() {
)     FB.init({
)         appId: '429967890466753', //App ID
)         xfbml: true,
)         version: 'v2.0'
));
//Definições de logon
);
});
```

```
// Carrega o JavaScript SDK
})(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s) [0];
  if (d.getElementById(id)) (return);
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/pt_BR/sdk.js";
  fjs.parentNode.insertBefore(js, fjs);
})(document, 'script', 'facebook-jssdk'));
```

**Figura 153**



Essa é a função padrão de carregamento do Facebook JavaScript SDK. Ela é fornecida pela documentação oficial do Facebook, disponível na seguinte URL:

<https://developers.facebook.com/docs/javascript/quickstart/v2.0>

Neste código apenas é indicado o local do SDK e o idioma a ser utilizado. O JavaScript SDK está disponível em todos os idiomas suportados pelo Facebook. Observe que na linha de comando a seguir indicamos o idioma e o país (pt\_BR).

```
js.src = "//connect.facebook.net/pt_BR/all.js";
```

Observe que no código fornecido pela documentação do Facebook o idioma padrão é o inglês americano (código en\_US).

## ***Open Graph (Graph API)***

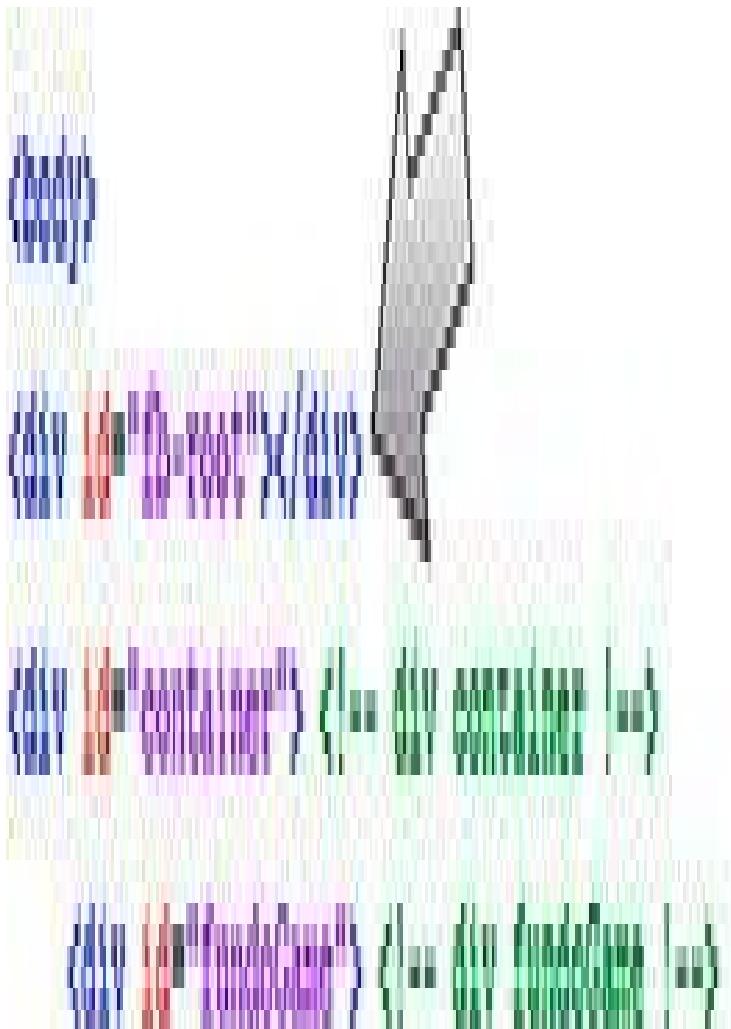
O método FB.init permite o acesso ao Open Graph do Facebook, que é a forma como o Facebook exibe e compartilha informações em sua página. Com o Open Graph é possível localizar informações, exibi-las e compartilhá-las.

O método FB.init somente dá acesso ao Open Graph, mas não exibe os dados. Para exibi-los o método utilizado deverá ser o FB.api, que veremos mais à frente.

No arquivo index.html deve ser criada uma div com o nome de fb-root, onde serão exibidas as mensagens do Facebook. Mesmo que você não deseje exibir informações nessa div, a sua criação é aconselhada pela documentação do Facebook.

1. Pressione as teclas Ctrl + S para salvar as alterações no arquivo js.js.
2. Abra o arquivo index.html e insira o código indicado a seguir após a tag <body>.

```
<div id="fb-root"></div>
```



### **Figura 154**

3. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Finalizamos o código de conexão com o Facebook SDK. O próximo passo é inserir o código que vai realizar a conexão do jogo com o Facebook.

Para que o usuário do Facebook possa utilizar o jogo, ele deve ter instalado o jogo em sua conta pessoal, que então ficará disponível no painel Aplicativos do Facebook (Figura 155).

## APLICATIVOS



Jogos

4



Hei War



Angry Birds Star Wars



Rescue your friends



Invacão do Planeta X



Observatório da Seleção



MeuFlashApp



Neste dia



Feed de jogos

20+

## **Figura 155**

4. Abra o arquivo js.js.

5. Digite o código a seguir após o comentário “//Definições de logon”, como indicado na Figura 156:

```
//Verifica se o usuário está logado
FB.getLoginStatus(function(response) {
  if (response.status === 'connected') {
    accessToken = response.authResponse.accessToken;
  } else if (response.status === 'not_authorized') {
    //Usuário logado ao Facebook, mas não registrado
    logar();
  } else {
    window.top.location ='https://www.facebook.com';
  }
})
```

});

```
//Conexão com o Facebook
] window.fbAsyncInit = function() {
} FB.init({
  appId: '238209999703632', //App ID
  xfbml: true,
  version: 'v2.0'
});

//Definições de logon

//Verifica se o usuário está logado
] FB.getLoginStatus(function(response) {
} if (response.status === 'connected') {
  accessToken = response.authResponse.accessToken;

} else if (response.status === 'not_authorized') {
  //Usuário logado no Facebook, mas não registrado
  logar();

} else {

  window.top.location = 'https://www.facebook.com';
}

}};

// Carrega o JavaScript SDK
] (function(d, s, id){
```

**Figura 156**



## Método FB.getLoginStatus

O método FB.getLoginStatus verifica o status do logon do usuário.

Observe que na definição do método indicamos a seguinte linha de código.

```
FB.getLoginStatus(function(response)
```

Inicia o método FB.getLoginStatus e executa uma função que terá como parâmetro da resposta a variável response.

```
if (response.status === 'connected')
```

Caso a propriedade status da variável response retorne como 'connected', indicando que o usuário está logado ao Facebook e registrado ao aplicativo, o token de acesso, que é uma chave eletrônica de acesso do aplicativo com o Facebook, será armazenado na variável accessToken, e a identificação do usuário será armazenada na variável accessToken pela linha de comandos a seguir.

```
accessToken = response.authResponse.accessToken;
```

A chave de acesso e a identificação do usuário são necessárias em alguns métodos que veremos mais à frente.

Caso a propriedade status seja retornada com o valor ‘not\_authorized’, indicado pela linha de comando a seguir,

```
else if (response.status === 'not_authorized')
```

a função logar() será executada.

```
logar();
```

O status ‘not\_authorized’ será retornado caso o usuário esteja logado ao Facebook, mas não tenha autorizado o aplicativo em sua conta pessoal.

Caso o usuário não esteja conectado ao Facebook, indicado pelas linhas de comando a seguir,

```
else {  
  window.top.location ='https://www.facebook.com';
```

a página atual será redirecionada ao Facebook para que ele possa efetuar o logon.

O próximo passo é criar a função logar(), que irá instalar o jogo na conta pessoal do usuário do Facebook.

1. Digite o código a seguir no local indicado na Figura 157.

```
//Função Logar
```

```
function logar() {  
  
    var oauth_url = 'https://www.facebook.com/dialog/oauth/';  
  
    oauth_url += '?client_id=429967890466753'; //App ID  
  
    oauth_url += '&redirect_uri=' + 'http://localhost/facebook/'; //Endereço URL do app  
  
    oauth_url +=  
        '&scope=user_about_me,email,user_location,user_photos,publish_actions,user_t  
  
    window.top.location = oauth_url;
```

}

-

Não se esqueça de alterar o appID para o número do seu aplicativo criado no Fa



```
// Carrega o JavaScript SDK
(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) {return;}
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/pt_BR/sdk.js#xfbml=1";
  fjs.parentNode.insertBefore(js, fjs);
  (document, 'script', 'facebook-jssdk'));
});
```

### //Função Logout

```
function logout() {
  var search_url = 'https://www.facebook.com/dialog/logout/';
  search_url += 'client_id=232309998733432'; //App ID
  search_url += 'redirect_uri=' + 'http://localhost/facebook/'; //Endereço URL do app
  search_url += 'scope=user_about_me,email,user_location,user_photos,publish_actions,user_birthday,user_friends';
  window.top.location = search_url;
}
```

**Figura 157**



## Caixa de diálogo OAuth

Na função `logar()` utilizamos a caixa de diálogo do Facebook OAuth, que é acessada pelo link <https://www.facebook.com/dialog/oauth/>. Utilizando este link, o Facebook exibirá automaticamente a caixa de diálogo de conexão do aplicativo.

Vamos identificar os parâmetros utilizados.

```
var oauth_url = 'https://www.facebook.com/dialog/oauth/';
```

Criamos uma variável que inicialmente recebeu o link de acesso à caixa de diálogo OAuth.

```
oauth_url += '?client_id=429967890466753';
```

Adicionamos à variável `oauth_url` o parâmetro `client_id` com o número do ID do aplicativo. Altere a numeração para o appID do seu aplicativo.

```
oauth_url += '&redirect_uri=' + http://localhost/facebook/'
```

Adicionamos à variável `oauth_url` o parâmetro `redirect_uri`, onde deve ser

indicado o endereço URL do aplicativo.

Chamando o método FB.login, é solicitado ao usuário que ele se registre ao aplicativo através da caixa diálogo OAuth.

Por padrão, chamando o método FB.login, o aplicativo irá autenticar o usuário apenas com as permissões básicas. Através da propriedade scope é possível indicar quais as permissões de acesso que o aplicativo terá na conta do usuário. Veja a seguir as principais permissões.

**email – Acesso ao e-mail do usuário.**

**user\_likes – Acesso aos likes do usuário, ou seja, o aplicativo terá acesso ao registro das páginas que o usuário curtiu.**

**user\_about\_me – Acesso à biografia do usuário.**

**user\_location – Acesso à localização atual do usuário.**

**user\_photos – Acesso ao álbum de fotos do usuário.**

**publish\_actions – O aplicativo terá permissão de publicar no mural do usuário.**

**user\_birthday** – Aniversário do usuário.

**user\_friends** – Permite acesso à lista de amigos do usuário, entre outras informações, como fotos etc. É importante saber que somente será possível receber informações dos amigos que possuírem o mesmo jogo instalado em sua conta no Facebook.

Vamos testar o funcionamento do código.

1. Pressione as teclas Ctrl + S para salvar as alterações.
2. Certifique-se de que o aplicativo XAMPP está em execução.
3. Certifique-se de que o módulo Apache do XAMPP está em execução.
4. Abra o Google Chrome.
5. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.

▪

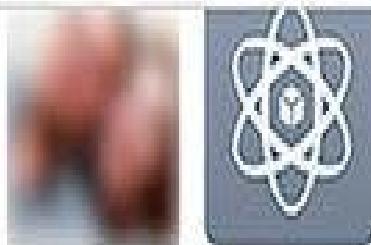
Caso você não esteja logado em sua conta do Facebook, a página de login será exibida.

Observe que a janela de autorização de instalação do aplicativo será exibida:



The following permissions have not been approved for use:  
`user_about_me`, `user_location`, `user_photos`, `publish_actions`,  
`user_birthday`, `user_likes`, `read_friendlists` e `user_activities`.

If you make your app public, they will not be shown to people using your  
app. Submit them for review or [saiba mais](#).



**Resgate** receberá as seguintes informações: perfil público,  
endereço de e-mail, listas de amigos personalizadas, aniversário,  
cidade atual, fotos, descrição pessoal e curtidas.

[Editar as informações fornecidas por você](#)

Isso não permite que o aplicativo publique no Facebook.

[Cancelar](#)

[OK](#)

## **Figura 158**

Observe que uma mensagem de alerta é exibida na janela. Recentemente o Facebook alterou a sua política de utilização das permissões. Algumas permissões, como por exemplo, exibir o nome ou fotos dos amigos do usuário, só serão concedidas quando o jogo estiver online (em um servidor web). Também só será possível exibir as fotos dos amigos que possuírem o mesmo aplicativo instalado em sua conta, ou seja, serão exibidas apenas as fotos dos amigos que possuírem o jogo Resgate instalado em sua conta.

Para que o Facebook libere algumas permissões específicas é necessário enviar o aplicativo para análise.

Clicando no botão “OK”, o usuário estará ciente de que o jogo poderá receber as informações especificadas.

6. Clique no botão “OK”.

Uma segunda janela de permissão será exibida, indicando que o jogo poderá realizar publicações no mural do usuário.



Resgate gestana de publicar no Facebook para você.  
Com quem você deseja compartilhar estas publicações?

 Amigos ▾

---

Agora não

OK

### **Figura 159**

7. Clique no botão “OK”.

Observe que o jogo será executado.

Com as permissões consentidas, a partir de agora é possível ter acesso ao OpenGraph do Facebook. No próximo capítulo exibiremos informações do usuário no jogo.

## **Capítulo 23**

### **Exibindo informações do jogador**

Neste capítulo exibiremos o nome do jogador na div fundoGame. O nome do jogador será obtido pelo acesso ao Open Graph do Facebook.

## ***Open Graph***

O Open Graph é a forma como o Facebook exibe e compartilha informações em sua página. Tendo acesso ao Open Graph pelo método FB.init, é possível localizar informações, exibi-las e compartilhá-las.

No capítulo anterior recebemos a autorização para ter acesso ao Open Graph. Vamos digitar um código de teste para exibir informações do usuário.

1. Abra o arquivo js.js e digite o código a seguir no local indicado pela Figura 160.

```
FB.api('/me', function(info) {  
  console.log(info);  
});
```

```
//Definições de logon

//Verifica se o usuário está logado
FB.getLoginStatus(function(response) {
    if (response.status === 'connected') {
        accessToken = response.authResponse.accessToken;

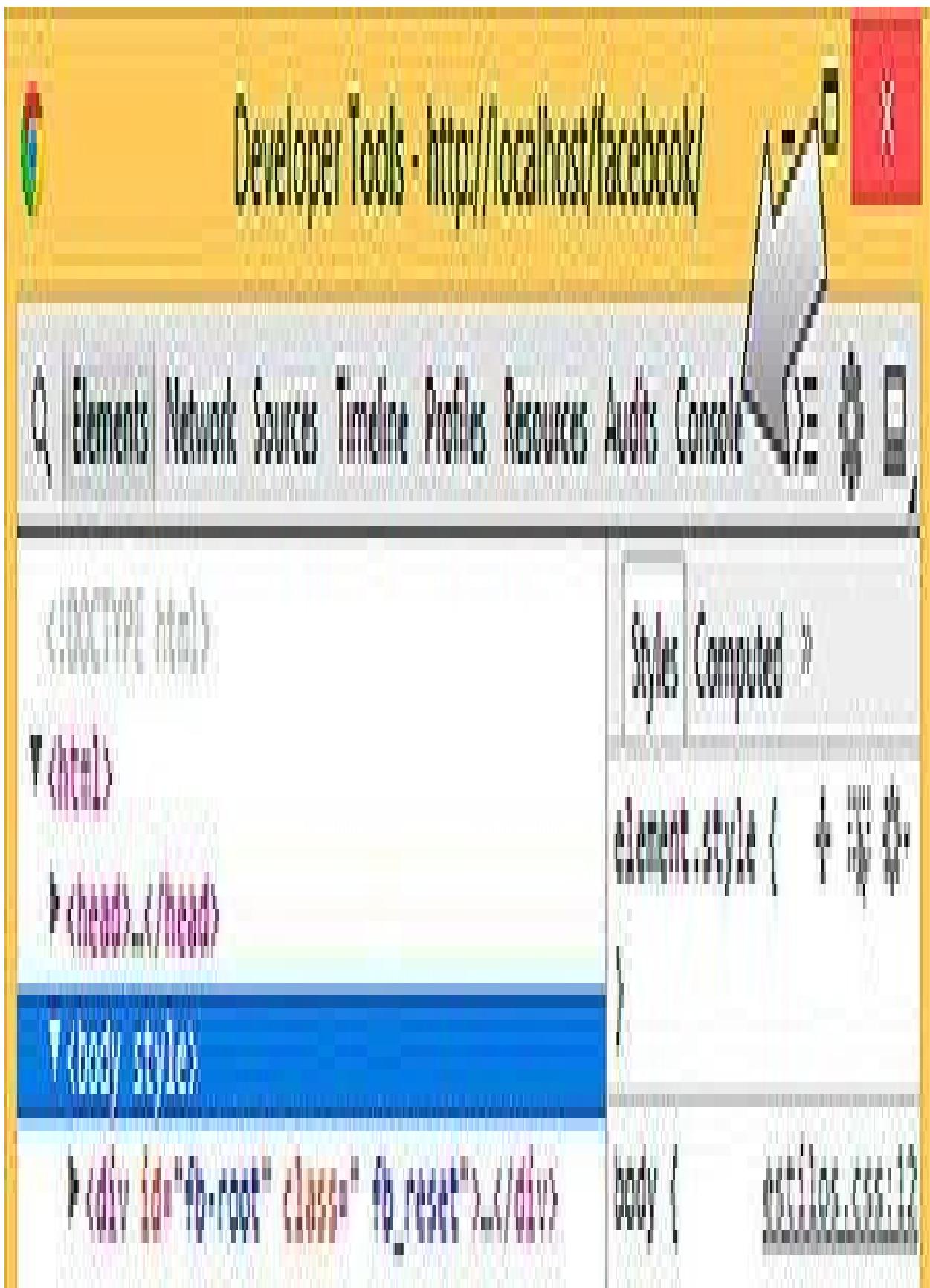
        FB.api('/me', function(info) {
            console.log(info);
        });
    } else if (response.status === 'not_authorized') {
        //Usuário logado no Facebook, mas não registrado
```

## **Figura 160**

2. Pressione as teclas Ctrl + S para salvar as alterações.
3. Abra o Google Chrome.
4. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.

Observe que nenhuma mensagem será exibida diretamente no jogo. O comando `console.log` exibe informações no console do browser.

5. Para ter acesso ao console do Google Chrome pressione as teclas Ctrl + Shift + I.
6. Observe que a janela Developer Tools será exibida. Clique na opção “Console” (Figura 161).



### **Figura 161**

7. Dê um clique sobre a opção “Object” exibida no console:

<top frame>

```
▶ Object {id: "1420179102", name: "Denilson Bonotti",  
  last_name: "Bonotti", link: "https://www.facebook.com/1420179102",  
  ...}
```

### **Figura 162**

Veja que as suas informações cadastradas no Facebook serão exibidas:

```
  ▷ Object {id: "1428179182", name: "Denilson Bonatti", first_name: "Denilson", l
    facebook.com/denilsonbonatti"} ⓘ
      bio: "Desenvolvedor e escritor de material didático sobre tecnologia. Desig
      birthday: "09/14/1979"
    ▷ education: Array[2]
      email: "denilsonbonatti@gmail.com"
    ▷ favorite_teams: Array[1]
      first_name: "Denilson"
      gender: "male"
      id: "1428179182"
      last_name: "Bonatti"
      link: "https://www.facebook.com/denilsonbonatti"
      locale: "pt_BR"
    ▷ location: Object
      name: "Denilson Bonatti"
```

### **Figura 163**

Todas essas informações exibidas no console podem também ser exibidas no jogo. Vamos realizar um exemplo prático.

Inicialmente vamos criar uma div para exibir o nome do jogador na div fundoGame.

8. Digite o código a seguir dentro da função start no local indicado na Figura 164.

```
$("#fundoGame").append("<div id='piloto'></div>");
```

```
function start() { // Início da função start()
    $('#inicio').hide();
    $('#fundoGame').append("<div id='jogador' class='animal1'></div>");  

    $('#fundoGame').append("<div id='inimigo1' class='animal2'></div>");  

    $('#fundoGame').append("<div id='inimigo2'></div>");  

    $('#fundoGame').append("<div id='amigo' class='animal3'></div>");  

    $('#fundoGame').append("<div id='placar'></div>");  

    $('#fundoGame').append("<div id='energia'></div>");  

    $('#fundoGame').append("<div id='piloto'></div>");
```

## **Figura 164**

O próximo passo é criar a formatação que será utilizada por essa nova div.

9. Abra o arquivo estilos.css e digite o código a seguir no final do arquivo.

```
#piloto {  
width: 350px;  
height: 35px;  
position: absolute;  
left: 30px;  
top: 8px;  
text-align: left;  
}
```

10. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Agora vamos criar o código que exibirá o nome do jogador.

11. Abra novamente o arquivo js.js e digite o código a seguir no local indicado na Figura 165.

```
//Exibe nome do Piloto

FB.api('/me', function(info) {
  var nome=info.first_name;
  $("#piloto").html("<h2> Piloto: " + nome + "</h2>");
});
```

```
function start() { // Início da função start()
  //Exibe nome do Piloto
  FB.api('/me', function(info) {
    var nome=info.first_name;
    $("#piloto").html("<h2> Piloto: " + nome + "</h2>");
  });
}

$("#inicio").hide();
```

**Figura 165**



Observe que, pelo código, estamos armazenando na variável nome a propriedade first\_name obtida pela função info. Na função info estamos utilizando o método FB.api para ter acesso às informações do usuário.

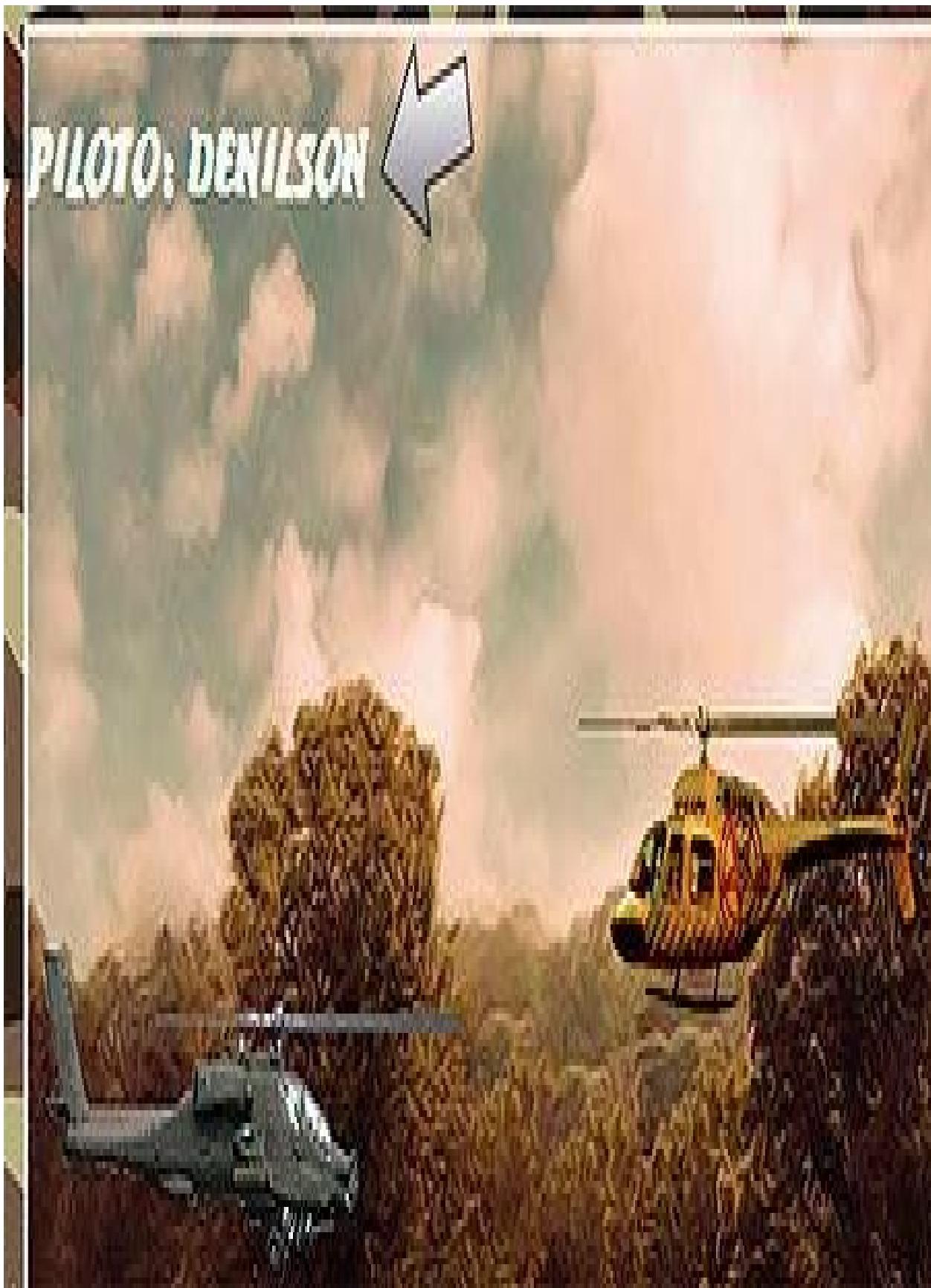
```
var nome=info.first_name;  
  
$("#piloto").html("<h2> Piloto: " + nome + "</h2>");
```

Veja na Figura 165 que a propriedade first\_name exibe o primeiro nome do usuário. A seguir o conteúdo da variável nome será exibido na div piloto.

12. Pressione as teclas Ctrl + S para salvar as alterações.
13. Abra o Google Chrome.
14. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.

Observe que o nome do jogador será exibido na div piloto (Figura 166).

**PILOTO: DENILSON**



## **Figura 166**

As consultas executadas a seguir servirão apenas como referência, pois seus resu

## **Capítulo 24**

## ***Facebook Query Language (FQL)***

O Facebook Query Language, ou FQL, utiliza uma sintaxe parecida com o SQL para consultar dados no Facebook. O FQL fornece algumas funcionalidades avançadas não disponíveis no método FB.ini. Por exemplo, para obter a foto principal do perfil do Facebook do jogador é necessário realizar uma consulta FQL.

Ao contrário do SQL, uma consulta FQL pode ser feita em uma tabela por vez, e as consultas não podem referenciar variáveis externas.

Segue a sintaxe do FQL:

```
SELECT [atributo1, atributo2, ...] FROM [tabela] WHERE [condições]
```

Para, por exemplo, encontrar o nome do usuário do jogo o código seria:

```
SELECT name FROM user WHERE uid = me()
```

Aqui estamos selecionando o campo name da tabela user, onde o campo uid seja igual a me(). A propriedade me() é utilizada para capturar o id do usuário logado no jogo.

## ***Graph API Explorer***

No Graph API Explorer é possível realizar testes de consultas FQL e demais funções do Facebook. Iremos utilizá-lo para obter respostas das consultas que iremos realizar.

1. Abra o Google Chrome e acesse a URL a seguir:

<https://developers.facebook.com/tools/explorer>

Para realizar testes utilizando o Graph API Explorer, inicialmente é necessário indicar as permissões de acesso. Como temos o jogo “Resgate” vinculado à conta do Facebook, vamos utilizar as permissões desse jogo.

2. Dê um clique na guia de seleção do botão “Aplicativo” (Figura 167).

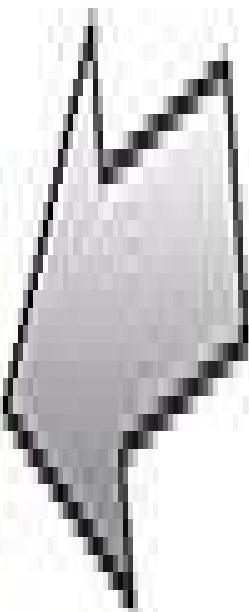


**Figura 167**

3. Selecione o aplicativo Resgate.
4. Dê um clique na opção “FQL Query” (Figura 168).



FQL Query



https://graph.facebook.com/me/friends

Some notes about the Graph API syntax:

## **Figura 168**

Vamos realizar a primeira consulta.

5. Digite no campo indicado na Figura 169 o código:

```
SELECT uid2 FROM friend WHERE uid1 = me()
```



### **Figura 169**

6. Clique no botão “Enviar”.

No código anterior, selecionamos o campo uid2 (número de id) da tabela friend (amigo). O registro consultado será o do usuário logado uid1 = me().

Observe que o Facebook exibe o campo id2 de todos os amigos do usuário:

```
{  
  "data": [  
    {  
      "uid2": "501796289"  
    },  
    {  
      "uid2": "538126426"  
    },  
    {  
      "uid2": "540122820"  
    },  
    {  
      "uid2": "570294475"  
    },  
    {  
      "uid2": "591578511"  
    },  
    {  
      "uid2": "601577562"  
    },  
    {  
      "uid2": "601577562"  
    }]  
}
```

## **Figura 170**

Caso você queira, por exemplo, consultar nome, likes, música ou fotos dos amigos de um determinado usuário, será necessário realizar uma consulta na tabela user e uma subconsulta na tabela friend.

7. Vamos realizar um exemplo: digite o código a seguir e clique no botão “Enviar”.

```
SELECT name FROM user WHERE uid IN(SELECT uid2 FROM friend  
WHERE uid1 = me())
```

Observe que os nomes dos amigos do usuário serão exibidos.

Outro exemplo: vamos consultar o nome do usuário, a foto principal do perfil e o tipo de música que ele gosta.

8. Digite o código a seguir e clique no botão “Enviar”:

```
SELECT first_name, pic_square, music FROM user WHERE uid IN (SELECT  
uid2 FROM friend WHERE uid1 = me())
```

Observe que o link da imagem principal do perfil do usuário será exibido (Figura 171).



THE BIRDS OF THE SOLOMON ISLANDS

"pic\_gamer": "https://fedoraproject.org/wiki/Project:Fedora\_gaming/2014\_q3#pic\_gamer",

"music"? "My So-So-So - Lynyrd Skynyrd Tribute", Scott, Florida, For Sale - show your music, Tracy James, Jimmy Buffet, or Heartbreak, Phillip S. Jennings & the Claypals, DR 528, Official Heartbreak, Phillip Jennings, Idaho, The White Buffalo, George White Records, South American, Double Agent, Bruce Springsteen, Joe Bonamassa, Van Halen, Alice Cooper, ZZ Top, David Lee, Tedeschi Trucks Band, Tom Petty, Arnel Pineda, Metallica 88 - 100.1, A Metal Rock, Clarence Carter, Sacramento 11, Augustana, Nickelback, Gary Stewart, Tedeschi Trucks, Skynyrd, Charlie Starns, Blackberry Smoke, The Black Crowes, Bruce L. On Groovin' Up, Martin Zandman, Annihilator, 30 Seconds, Iron Maiden, Megadeth, Bon Jovi, Metallica, Megadeth, Rock Mafia, Gary Busey, Black Sabbath, Disturbed, Scott Stapp, Hellzapoppin, Le Prince, MySpace Release, Judas Priest, Judas Priest, Judas Priest, Megadeth Christian, 2007, METALLICA, Cross Country, Queen, MUSE, Andrew Lloyd, The Classics 2007, Angus Young, Jimi Hendrix, Bob Dylan, Foster The People, Clapton, AC/DC, Lynyrd Skynyrd, Lynyrd Skynyrd, Cobain and Company, METALLICA, Andrew Lloyd, Black Label Society, Sepultura, METALLICA 2007, Alter Bridge, Black Label Society

## **Figura 171**

Para saber todas as propriedades que podem ser consultadas na tabela user visite o link:

<https://developers.facebook.com/docs/reference/fql/user>

Para conhecer todas as tabelas disponíveis para consulta utilizando o FQL acesse o link:

<https://developers.facebook.com/docs/reference/fql/>

No jogo que estamos desenvolvendo iremos utilizar as seguintes consultas:

SELECT pic\_square FROM user WHERE uid = me()

Essa consulta retorna a foto principal do usuário no Facebook.

SELECT first\_name, pic\_square FROM user WHERE uid IN (SELECT uid2 FROM friend WHERE uid1 = me())

A consulta anterior retorna o nome e a foto principal dos amigos do usuário.

Vamos ao código que exibirá a foto do usuário no jogo. Inicialmente vamos criar o código CSS e criar as novas divs que irão posicionar os novos elementos no jogo.

1. Abra o arquivo estilos.css e digite o código a seguir no final do arquivo.

```
#foto1 {  
width: 30px;  
height: 30px;  
position: absolute;  
background-size:30px;  
left: 35px;  
text-align: left;  
border-color: #FFF;  
border-style: solid;  
border-width: thin;  
}  
}
```

```
#foto2 {  
width: 30px;
```

```
height: 30px;  
background-size:30px;  
position: absolute;  
border-color: #FFF;  
border-style: solid;  
border-width: thin;  
}
```

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar as divs no arquivo js.js.

3. Abra o arquivo js.js e digite o código a seguir no local indicado na Figura 172.

```
$("#fundoGame").append("<div id='foto1'></div>");
```

```
$("#fundoGame").append("<div id='foto2'></div>");
```

```
$("#fundoGame").append("<div id='jogador' class='animal1'></div>");  
$("#fundoGame").append("<div id='inimigo1' class='animal2'></div>");  
$("#fundoGame").append("<div id='inimigo2'></div>");  
$("#fundoGame").append("<div id='anigo1' class='animal3'></div>");  
$("#fundoGame").append("<div id='placar'></div>");  
$("#fundoGame").append("<div id='energia1'></div>");  
$("#fundoGame").append("<div id='piloto'></div>");  
$("#fundoGame").append("<div id='foto1'></div>");  
$("#fundoGame").append("<div id='foto2'></div>");
```

## Figura 172

O próximo passo é criar o código que exibirá a foto na div criada. Vamos utilizar o método FB.api, que irá executar uma consulta FBQ e exibir a imagem na div foto1.

4. Digite o código a seguir no local indicado na Figura 173.

```
//Exibe foto do Piloto
```

```
FB.api(  
{  
  method: 'fql.query',  
  query: 'SELECT pic_square FROM user WHERE uid = me()',  
,  
function(data) {  
  var dados=data;  
  var foto=dados[0].pic_square;  
  $('#foto1').css("background-image", 'url("' + foto + '")');
```

});

```
//Exibe nome do Piloto

FB.api('/me', function(info) {
  var nome=info.first_name;
  $("#piloto").html("<h2> Piloto: " + nome + "</h2>");

});
```

```
//Exibe foto do Piloto

FB.api(
  {
    method: 'fql.query',
    query: 'SELECT pic_square FROM user WHERE uid = me()',
  },
  function(data) {
    var dados=data;
    var foto=dados[0].pic_square;

    $("#fotol").css("background-image", 'url("' + foto + '")');

  });

```

**Figura 173**



No método FB.api indicamos pela propriedade method que iremos realizar uma consulta FBQ.

Na propriedade query indicamos a consulta que será realizada. Nesse exemplo, estamos consultando o campo pic\_square da tabela user.

```
FB.api(
```

```
{
```

```
  method: 'fql.query',
```

```
  query: 'SELECT pic_square FROM user WHERE uid = me()',
```

A resposta a essa consulta será retornada no parâmetro data.

```
  function(data) {
```

Criamos uma variável com o nome de foto que irá receber os dados da propriedade pic\_square.

```
    function(data) {
```

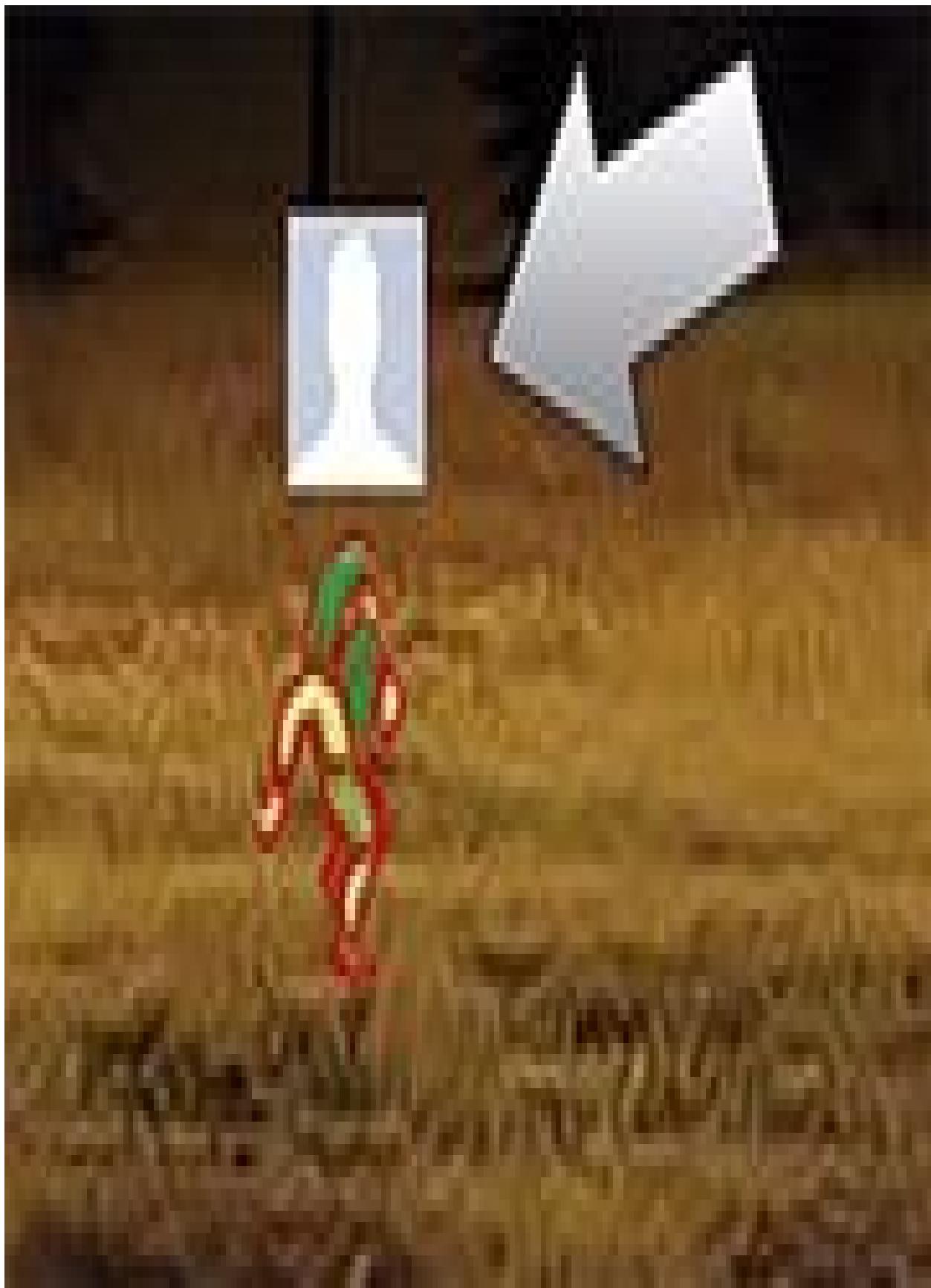
```
var dados=data;
```

```
var foto=dados[0].pic_square;
```

A seguir alteramos a propriedade CSS background-image da div foto1 com o valor recebido na variável foto.

```
$("#foto1").css("background-image", 'url("' + foto + '")');
```

O próximo passo é criar uma função com o nome exibeAmigos que exibirá as fotos dos amigos do usuário do jogo sobre a div amigo (Figura 174).



## **Figura 174**

5. Digite o código a seguir no local indicado na Figura 175.

```
//Exibe fotos dos amigos
```

```
function exibeAmigos() {
```

```
    FB.api(
```

```
{
```

```
    method: 'fql.query',
```

```
    query: 'SELECT pic_square FROM user WHERE uid IN(SELECT uid2 FROM friend WHERE uid1 = me())',
```

```
},
```

```
function(data) {
```

```
    var dados=data;
```

```
    var total=dados.length;
```

```
    var amigo1=parseInt(Math.random() * total);
```

```
    var foto2=dados[amigo1].pic_square;
```

```
$("#foto2").css("background-image", 'url("' + foto2 + "')');
```

```
});
```

```
}
```

```
//Exibe foto do Piloto

FB.api(
{
  method: 'fql.query',
  query: 'SELECT pic_square FROM user WHERE uid = me()',
},
function(data) {
  var dados=data;
  var foto=dados[0].pic_square;

  $("#foto1").css("background-image", 'url("'+ foto +'")');
});

```

```
//Exibe fotos dos amigos

function exibeAmigos() {

FB.api(
{
  method: 'fql.query',
  query: 'SELECT pic_square FROM user WHERE uid IN(SELECT uid2 FROM friend WHERE uid1 = me())',
},
function(data) {
  var dados=data;
  var total=dados.length;
  var amigol=parseInt(Math.random() * total);
  var foto2=dados[amigol].pic_square;

  $("#foto2").css("background-image", 'url("'+ foto2 +'")');
});
}
```

**Figura 175**



Nesta função exibiremos os amigos do usuário de forma aleatória, ou seja, cada vez que essa função for executada, um amigo diferente do usuário será exibido na div foto2.

No método FB.api indicamos pela propriedade method que iremos realizar uma consulta FBQ. Na propriedade query indicamos a consulta que será realizada. Nesse exemplo, estamos consultando o campo pic\_square da tabela user dos amigos do usuário.

```
method: 'fql.query',  
query: 'SELECT pic_square FROM user WHERE uid IN(SELECT uid2 FROM friend WHERE uid1 = me())',
```

A resposta a essa consulta será retornada no parâmetro data.

```
function(data) {
```

Para exibir os amigos aleatoriamente, criamos uma variável com o nome de total que irá receber o total de amigos do usuário. Como os dados são armazenados na variável em forma de uma array, para saber o total de amigos utilizamos a função length.

```
var total=dados.length;
```

Para exibir um amigo aleatoriamente, criamos uma variável com o nome amigo1 que, utilizando a função Math.random, irá selecionar um amigo aleatoriamente pelo código a seguir:

```
var amigo1=parseInt(Math.random() * total);
```

Para finalizar, criamos uma variável com o nome de foto2 para receber a propriedade pic\_square do amigo selecionado aleatoriamente e exibi-la na div foto2, através do código a seguir:

```
var foto2=dados[amigo1].pic_square;
```

```
$("#foto2").css("background-image", 'url("' + foto2 + '")');
```

O próximo passo é criar uma função com o nome de atualizaFotos, que irá atualizar a posição das divs foto1 e foto2 no jogo.

Inicialmente chamaremos uma função com o nome de atualizaFotos dentro do game loop do jogo.

6. Digite o código a seguir no local indicado pela Figura 176.

```
atualizaFotos();
```

```
function loop() {  
    movefundo();  
    movejogador();  
    moveinimigo1();  
    moveinimigo2();  
    moveamigo();  
    colisao();  
    placar();  
    energia();  
    atualizaFotos();  
}  
// Fim da função loop()
```

## **Figura 176**

7. O próximo passo é criar a função atualizaFotos. Digite o código a seguir no local indicado na Figura 177.

```
//Função que atualiza as posições das fotos

function atualizaFotos() {

    var topo1 = parseInt($("#jogador").css("top"));

    $("#foto1").css("top",topo1+40);

    var topo2 = parseInt($("#amigo").css("top"));

    var esquerda = parseInt($("#amigo").css("left"));

    $("#foto2").css("top",topo2-30);

    $("#foto2").css("left",esquerda);

} // Fim da função atualizaFotos()
```

```
    } // Fim da função gameOver();  
  
    //Função que atualiza as posições das fotos  
    function atualizaFotos() {  
  
        var topo1 = parseInt($("#jogador").css("top"));  
        $("#foto1").css("top", topo1+40);  
  
        var topo2 = parseInt($("#amigo").css("top"));  
        var esquerda = parseInt($("#amigo").css("left"));  
        $("#foto2").css("top", topo2-30);  
        $("#foto2").css("left", esquerda);  
  
    } // Fim da função atualizaFotos()  
  
} // Fim da função start  
  
//Reinicia o Jogo
```

### **Figura 177**

A seguir a função exibeAmigos deve ser “chamada” no início do jogo, dentro da função start.

8. Digite o código a seguir no local indicado na Figura 178.

```
exibeAmigos();
```

```
function start() { // Início da função start()
```

```
  exibeAmigos();
```

```
  //Exibe nome do Piloto
```

```
  FB.api('/me', function(info) {
```

```
    var nome=info.first_name;
```

### **Figura 178**

9. O próximo passo é atualizar a função reposicionaAmigo, que deverá chamar a nova função exibeAmigos e exibir a div foto2. Digite o código a seguir no local indicado na Figura 179.

```
$("#foto2").show();
```

```
exibeAmigos();
```

```
/Reposiciona Amigo
```

```
function reposicionaAmigo() {
```

```
var tempoAmigo=window.setInterval(reposiciona6, 6000);
```

```
function reposiciona6() {
```

```
window.clearInterval(tempoAmigo);
```

```
tempoAmigo=null;
```

```
if (fimdejogo==false) {
```

```
$("#fundoGame").append("<div id='amigo' class='anim3'></div>");
```

```
$("#foto2").show();
```

```
exibeAmigos();
```

```
}
```

```
}
```

### **Figura 179**

10. Para finalizar, a div foto2 deverá ser ocultada nas colisões com a divs jogador e inimigo2. Digite o código a seguir nos locais indicados na Figura 180.

```
$("#foto2").hide();
```

```
// jogador com o amigo

if (colisao5.length>0) {
    $("foto2").hide();
    somResgate.play();
    salvos++;
    reposicionaAmigo();
    $("amigo").remove();
}

}
```

// Inimigo2 com o amigo

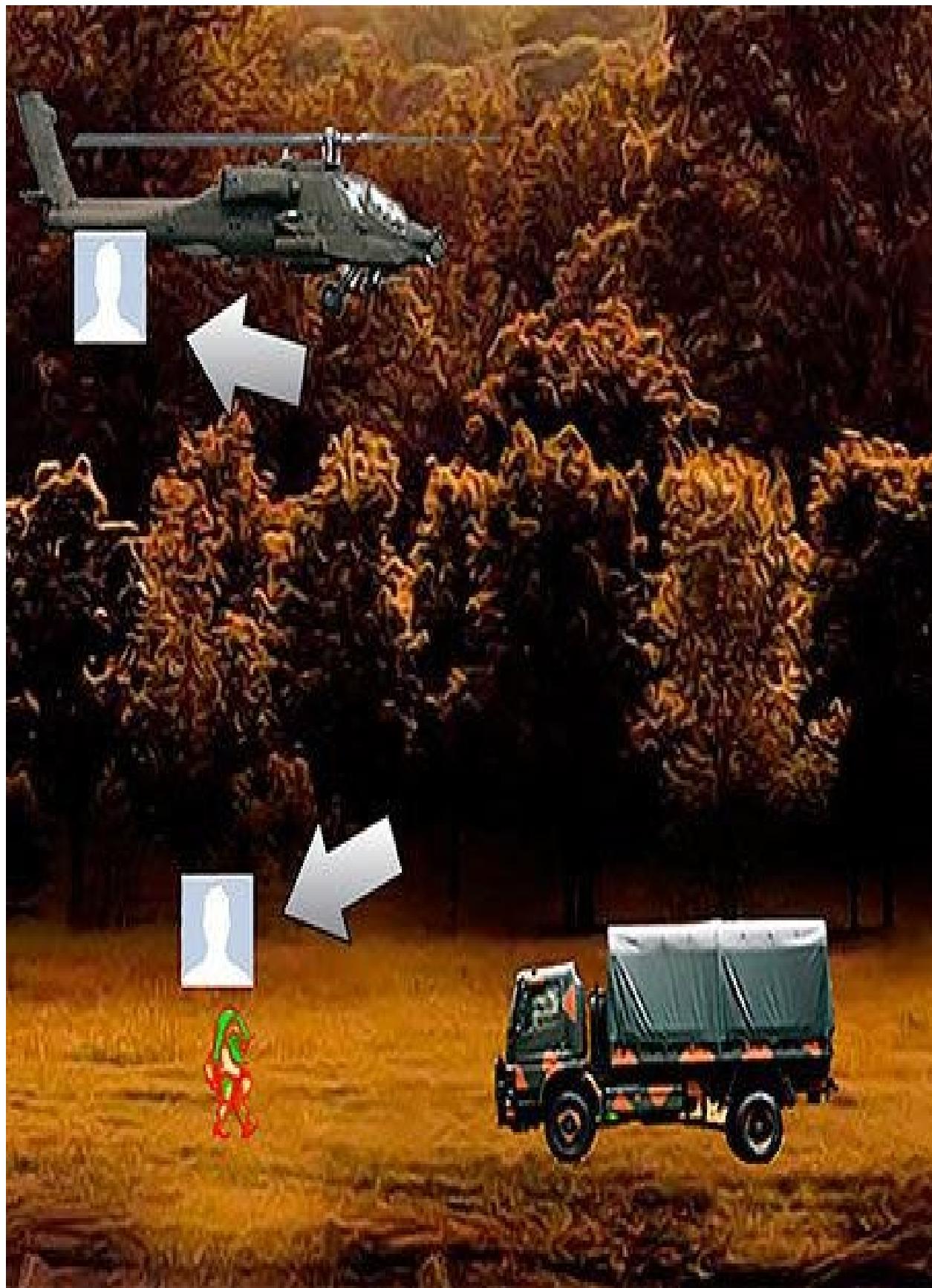
```
if (colisao6.length>0) {
    $("foto2").hide();
    perdidos++;
    amigoX = parseInt($("#amigo").css("left"));
    amigoY = parseInt($("#amigo").css("top"));
    explosao3(amigoX,amigoY);
    $("amigo").remove();
```

## **Figura 180**

Vamos testar o funcionamento do código.

11. Pressione as teclas Ctrl + S para salvar as alterações.
12. Abra o Google Chrome.
13. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.

Observe que a imagem do jogador será exibida normalmente na div foto1. A imagem do amigo na div foto2 somente será exibida quando o jogo estiver online (hospedado em um servidor web).



## **Figura 181**

Como você deve ter percebido, as divs foto1 e foto2 não são excluídas quando o jogo é finalizado. Para encerrar este capítulo vamos remover as divs foto1 e foto2 na função gameOver.

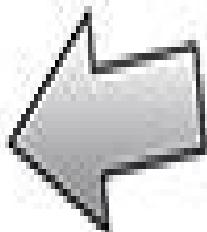
14. Digite o código a seguir no local indicado na Figura 182.

```
$("#foto1").remove();
```

```
$("#foto2").remove();
```

//Função GAME OVER

```
function gameOver () {  
    fimdejogo=true;  
    musica.pause();  
    somGameover.play();  
  
    window.clearInterval(jogo.timer);  
    jogo.timer=null;  
  
    $("#jogador").remove();  
    $("#inimigo1").remove();  
    $("#inimigo2").remove();  
    $("#amigo").remove();  
    $("#foto1").remove();  
    $("#foto2").remove();
```



### **Figura 182**

15. Pressione as teclas Ctrl + S para salvar as alterações e teste o jogo novamente.

## Capítulo 25

### Compartilhando a pontuação

Neste capítulo criaremos uma função que irá compartilhar a pontuação obtida pelo jogador no mural do Facebook.

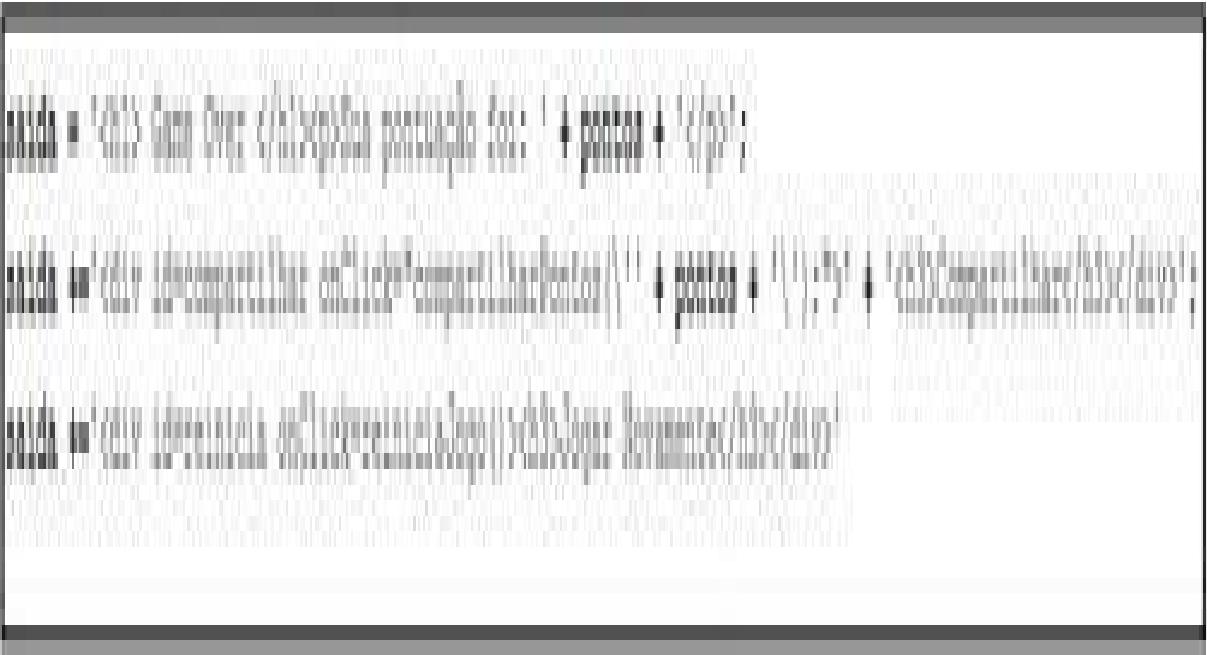
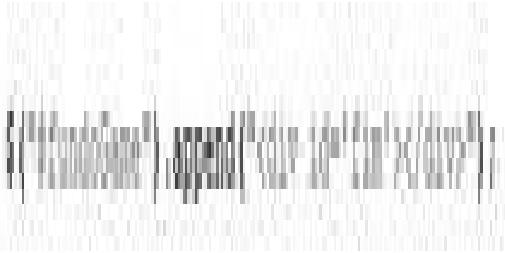
Inicialmente criaremos uma nova div dentro da div fim, onde exibiremos o texto “Compartilhar”. Caso o jogador clique sobre essa div, a função compatilharPontos será executada. A pontuação do jogador será compartilhada com seus amigos do Facebook.

1. No arquivo js.js, digite o código a seguir dentro da função gameOver no local indicado na Figura 183.

```
saída = '<h1> Game Over </h1><p>Sua pontuação foi: ' + pontos + '</p>';
```

```
saída += '<div id=compartilhar onClick="compatilharPontos(\'' + pontos + '\');">' + '<h3>Compartilhar</h3></div>';
```

```
saída += '<div id=reinicia onClick=reiniciaJogo()><h3>Jogar Novamente</h3></div>'
```



### **Figura 183**

O próximo passo é exibir o conteúdo da variável saída na div fim.

2. Selecione o código indicado na Figura 184.



### **Figura 184**

3. Pressione a tecla Delete para excluir o código selecionado. Em seu lugar digite o código a seguir:

```
$("#fim").html(saida);
```

O próximo passo é criar a função compartilharPontos.

## Utilizando o método FB.ui

Vamos utilizar o método FB.ui para exibir uma caixa de diálogo do Facebook. Observe que indicaremos na propriedade method o valor feed. Com tal valor, a caixa de diálogo exibida será “Publicar no mural”.

1. No arquivo js.js, digite o código a seguir no local no final do arquivo, fora da função start (Figura 185).

```
//Função que compartilha a pontuação do Jogador
function compartilharPontos(pontos) {

    var obj = {
        method: 'feed',
        redirect_uri: 'https://apps.facebook.com/resgate',
        picture: 'http://www.denilsonbonatti.com.br/livros/jogo1/imgs/icon.png',
        caption: 'Conteúdo do livro de desenvolvimento de jogos em HTML5',
        description: 'Eu consegui a seguinte pontuação: ' + pontos + '. Você consegue me
superar?'
    };
}
```

```
function callback(response) {
```

```
    reiniciaJogo();
```

```
}
```

```
    FB.ui(obj, callback);
```

```
}
```

```
//Reinicia o Jogo

function reiniciaJogo() {
    somGameover.pause();
    $('#fim').remove();
    start();
}

} //Fim da função reiniciaJogo
```

```
//Função que compartilha a pontuação do Jogador
function compartilharPontos(pontos) {

    var obj = {
        method: 'feed',
        redirect_uri: 'https://www.facebook.com/sharer/',
        picture: 'http://www.denilsonbonatti.com.br/livros/jogos/imgs/icon.png',
        caption: 'Conteúdo do livro de desenvolvimentos de jogos em HTML5',
        description:'Eu consegui a seguinte pontuação:' + pontos + '. Você consegue me superar?'
    };

    function callback(response) {
        reiniciaJogo();
    }

    FB.ui(obj, callback);
}
```

**Figura 185**



Note que inicialmente criamos um objeto:

```
var obj = {
```

que possui o atributo method:

```
method: 'feed',
```

Indicamos que o método será feed, utilizado para compartilhar informações no mural do usuário.

Em seguida temos:

```
redirect_uri: 'https://apps.facebook.com/resgate',
```

que indica a URL do aplicativo.

■ Substitua essa linha pela URL do seu aplicativo criado no Facebook. Em caso de

Em seguida temos a imagem que será utilizada na postagem. Ela deve estar obrigatoriamente online. Utilize o link indicado como exemplo.

picture: 'http://www.denilsonbonatti.com.br/livros/jogo1/imgs/icon.png'

Logo após, temos o atributo caption, que é utilizado para indicar o título do post.

caption: 'Conteúdo do livro de desenvolvimento de jogos em HTML5.'

Por último, temos que indicar o conteúdo da publicação pelo método description.

description:'Eu consegui a seguinte pontuação:' + pontos + '. Você consegue me superar?'

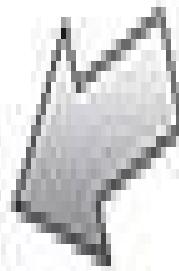
Após a postagem, a função reiniciaJogo será executada, reiniciando o jogo.

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
3. Abra o Google Chrome.
4. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.
5. Ao finalizar o jogo, clique no texto “Compartilhar”:

# GAME OVER

Sua pontuação foi 1250

**COMPARTILHAR**



**JOGAR NOVAMENTE**

### **Figura 186**

Veja que uma janela do Facebook será exibida para compartilhamento do texto indicado na função.

## Publicar no Facebook



### Minha Pontuação!!!



Conteúdo do livro de desenvolvimento de jogos em HTML5  
Eu consegui a seguinte pontuação: 1250. Você consegue me superar?



Compartilhar

Cancelar

### **Figura 187**

Clicando no botão “Compartilhar” o texto será exibido no mural do usuário:



Procurar pessoas, locais e coisas



Denilson Bonatti  
Editar perfil



Status



Adicionar fotos/video



Feed de Notícias



Mensagens

2



Eventos

1

DESENVOLVEDOR



Resgate Livro



Agora Vai



Rescue your friends



Resgate



Hell War



Invasão do Planeta X

GRUPOS



Autores do Buzzfeed... 4



TG-02-018 Jabotic... 2



VENDO & TROCO... 20+



Criar grupo

APLICATIVOS



Jogos



Angry Birds Star Wars



Wish You Were Here?



DengueVille



Xbox LIVE Social Exp...



Deezer



Denilson Bonatti

66 · 1 · 0 · 0



Minha Pontuação!!!

Eu consegui a seguinte pontuação: 1250. Você consegue me superar?

CONTEUDO DO LIVRO DE DESENVOLVIMENTO DE JOGO...



Escreva um comentário...

[Ver mais](#)

### **Figura 188**

O próximo passo é exibir os plugins sociais do Facebook.

## ***Plugins sociais***

Os plugins sociais do Facebook são utilizados para curtir e compartilhar informações de uma página web ou aplicativo. Utilizaremos os plugins sociais para melhorar a visibilidade do jogo.

Inicialmente vamos criar uma nova div onde serão exibidos os plugins. Essa div será exibida no carregamento da página e não quando o jogo iniciar. Sendo assim, vamos criar a div diretamente no código HTML.

1. Abra o arquivo index.html e digite o código a seguir no local indicado na Figura 189.

```
<div id="social"> <!--início da div social !-->
```

```
<h3>Este jogo foi produzido no livro Desenvolvimento de jogos em  
HTML5</h3>
```

```
<div class="fb-like" data-href="https://apps.facebook.com/resgate" data-  
send="true" data-width="755" data-show-faces="true" font="arial"></div>
```

```
<p class="textoSocial"> Gostou do Jogo? Deixe o seu comentário ou a sua  
sugestão!</p>
```

```
<div class="fb-comments" data-width="510" data-  
href="https://apps.facebook.com/resgate" data-num-posts="5"></div>
```

```
<div id="imagem">
```

```
</div>
```

```
</div> <!-- fim da div social !-->
```

```
para iniciar!! </p>
</div> <!-- Fim da div inicio !-->

</div> <!-- Fim da div fundoGame !-->
```

```
<div id="social"> <!---início da div social !-->
<h3>Este jogo foi produzido no livro Desenvolvimento de jogos em HTML5</h3>
<div class="fb-like" data-href="https://apps.facebook.com/resgate" data-send="true" data-width="755" data-show-faces="true" font="arial"></div>
<p class="textoSocial"> Gostou do Jogo? Deixa o seu comentário ou a sua
sugestão!</p>
<div class="fb-comments" data-width="510" data-href="https://apps.facebook.com/resgate" data-num-posts="5"></div>
<div id="image">

</div> <!-- Fim da div social !-->
```

```
</div> <!-- Fim da div container !-->
```

**Figura 189**



Como o jogo está vinculado ao Facebook, basta nomear a div com a classe do plugin desejado para que ele seja exibido.

A classe fb-like exibe os botões de curtir e compartilhar.

```
<div class="fb-like" data-href="https://apps.facebook.com/resgate" data-send="true" data-width="755" data-show-faces="true" font="arial"></div>
```

Observe que os seguintes parâmetros foram indicados:

data-href="https://apps.facebook.com/resgate"

onde deve ser indicada a URL de referência do botão “Curtir”. Nesse exemplo, o que será curtido é o endereço URL do aplicativo.

■

Observe que estamos indicando o link do aplicativo. Altere o link do aplicativo

data-show-faces="true"

Como definimos o valor dessa propriedade como true, a foto principal do perfil dos usuários que curtirem o aplicativo será exibida. Vamos mostrar as imagens dos usuários, pois há espaço disponível na div social para isso. Para exibir o botão “Curtir” em locais de menor espaço, utilize o valor false.

data-width="510"

Indicamos a largura da caixa do botão em 510 pixels.

font="arial"

O parâmetro font indica a fonte a ser utilizada no plugin. As opções são “arial”, “lucida grande”, “segoe ui”, “tahoma”, “trebuchet ms” e “verdana”. Neste exemplo estamos utilizando a fonte arial.

A seguir temos uma div definida com a classe fb-comments. Essa classe exibe a caixa de comentários.

```
<div class="fb-comments" data-width="510" data-href="https://apps.facebook.com/resgate" data-num-posts="5"></div>
```

Observe que indicamos os seguintes parâmetros:

```
data-href= 'https://apps.facebook.com/resgate'
```

Indicamos a URL de referência do comentário como sendo a URL do jogo. Dessa forma, os comentários estão vinculados ao jogo.

```
data-width="510"
```

Largura da caixa do comentário. Nesse exemplo utilizamos 510 pixels.

```
data-num-posts="5"
```

Parâmetro utilizado para definir o número de comentários que serão exibidos. Neste exemplo indicamos cinco.

Finalizando o código, inserimos uma imagem dentro de uma div com o nome de imagem que, quando clicada, irá executar uma função chamada convidar.

```
<div id="imagem">
```

```

</div>
```

Na função convidar vamos criar o código que irá enviar as requisições para os amigos do usuário.

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é indicar a formatação CSS das divs criadas.

3. Abra o arquivo estilos.css e digite o código a seguir no final do arquivo:

```
#social {  
    width:940px;  
    padding:5px;  
    height:auto;  
    position:relative;  
    background-color:#eddeded;  
    border-color: #FFF;  
    border-style: solid;  
}  
}
```

```
#imagem {  
width:250px;  
height:292px;  
position:relative;  
border-color: #FFF;  
border-style: solid;  
}  
}
```

```
.textoSocial {  
font-family:arial;  
font-size:15px;  
color:#000000;
```

}

4. Pressione as teclas Ctrl + S para salvar as alterações.

A seguir vamos criar a função convidar no arquivo js.js.

5. Abra o arquivo js.js.

6. Digite o código a seguir no final do arquivo, no local indicado na Figura 190.

```
//Função convidar
```

```
function convidar() {
```

```
    FB.ui({
```

```
        method: 'apprequests',
```

```
        title: 'Resgate os seus Amigos do Facebook!!',
```

```
        message: 'Eu estou jogando o Game que eu criei. Você quer jogar também?'
```

```
    });
```

```
}
```

```
function callback(response) {  
  reiniciaJogo();  
}  
  
FB.ui(obj, callback);  
}
```

//Função convidar

```
function convidar() {  
  FB.ui({  
    method: 'apprequests',  
    title: 'Resgate os seus amigos do Facebook!!!',  
    message: 'Eu estava jogando o Game que eu criei. Você quer jogar também?'  
});  
}
```

### **Figura 190**

7. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
8. Abra o Google Chrome.
9. Digite a URL <http://localhost/facebook> e pressione a tecla Enter.

Observe que, quando o jogo é iniciado, os plugins sociais são exibidos (Figura 191).

**ESTE JOGO FOI PRODUZIDO NO LIVRO DESENVOLVIMENTO DE JOGOS EM HTML5**



Gostou do jogo? Deixe o seu comentário ou a sua sugestão!



### **Figura 191**

10. Clique na imagem “Convide seus amigos”.

Veja que a janela de solitações será exibida (Figura 192).



### Visualização da solicitação:



Eu estou jogando o Game que eu criei. Você quer jogar também?

Todos os amigos +

Procurar...



Débora Bonatti Boarini

Convidar



Mariana Dos Santos Trevisoli

Convidar



Rodrigo Bonatti

Convidar



Daniel Bonatti

Convidar



Rogério Bonatti

Convidar



Rodrigo Romão

Convidar

Não perguntar novamente antes de enviar solicitações deste aplicativo para esses amigos.

Concluir

**Figura 192**

## Capítulo 26

### Hospedando o jogo em um servidor web

Finalizada a fase de testes do jogo utilizando um servidor local, o próximo passo é hospedar o jogo em um servidor web para que ele possa ser acessado de qualquer local do mundo via Facebook.

Antes de hospedar os arquivos em um servidor web, devemos substituir os links vinculados ao endereço localhost para o endereço no servidor web.

1. Faça uma cópia da pasta facebook para o desktop ou outro local de sua preferência, onde iremos realizar as alterações.
2. Abra o arquivo js.js da pasta copiada.
3. Altere todos os códigos vinculados ao localhost para o endereço URL de um servidor web.

Veja o exemplo a seguir:

```
function logar() {  
  var oauth_url = 'https://www.facebook.com/dialog/oauth/';  
  oauth_url += '?client_id=1394884477440571'; //App ID
```

```
oauth_url += '&redirect_uri=' + 'https://minhapagina/facebook/'; //Endereço  
URL do app
```

Observe, no exemplo, que indicamos o endereço URL do servidor web utilizando uma conexão segura pelo protocolo HTTPS. Para o pleno funcionamento do jogo é necessária uma conexão segura. Para isso, o servidor web onde os arquivos do jogo serão hospedados deve oferecer esse recurso.

Alguns provedores de hospedagem oferecem uma conexão SSL compartilhada, não necessitando de um certificado de segurança específico para o endereço URL. A URL SSL pode ser configurada através do painel de controle do servidor web contratado. Caso você tenha um servidor web gratuito, talvez esse recurso não esteja disponível.

# DESENVOLVEDOR

denilsonbonatti [Alterar](#)

## URL SSL

<https://site.com.br/websiteseclaro.com>

### **Figura 193**

Após alterar todos os endereços do localhost para uma URL segura do servidor web, o próximo passo é enviar os arquivos para o servidor web.

## Utilizando o Filezilla

O próximo passo é enviar a pasta facebook do desktop para o servidor web. Para realizar o upload dos arquivos, utilizaremos o aplicativo Filezilla.

1. Execute o aplicativo Filezilla.
2. O próximo passo é criar uma conexão FTP com o servidor web. Dê um clique no menu Arquivo e selecione a opção “Gerenciador de Sites”.

Arquivo Editar Ver Transferir Fornecedor Marcadores Ajuda

Gerenciador de Sites...

CTRL+S

Copiar a conexão atual para o Gerenciador de Sites...

Nova aba

CTRL+T

Fechar aba

CTRL+W

**Figura 194**

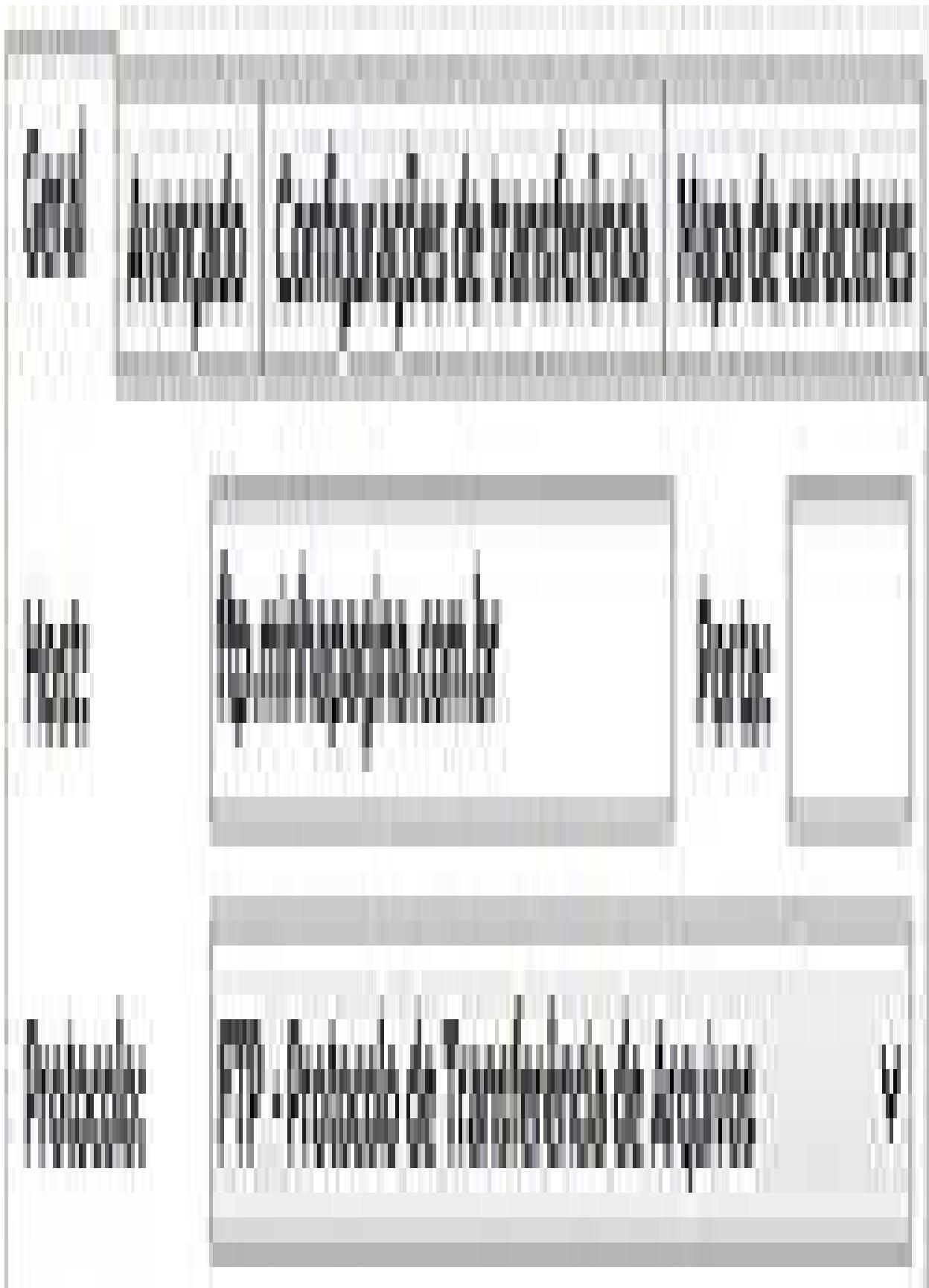
3. Dê um clique no botão “Novo Site”.
4. Nomeie esse novo site como facebook (Figura 195).

# Selección Entrada



**Figura 195**

5. Indique o endereço FTP do servidor web na opção “Host”.



### **Figura 196**

6. Dê um clique na guia de seleção da opção “Tipo de logon” e selecione “Normal”.
7. Digite o nome de usuário e a senha de acesso ao servidor web.



**Figura 197**

8. Clique no botão “Conectar”.
9. Após o Filezilla se conectar ao servidor web, dê um clique duplo na pasta public\_html ou www (dependendo da configuração do seu servidor web).

# Nome



..



.gems



.ssh



includes



logs



public\_html



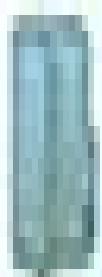
tmp



### **Figura 198**

10. Clique com o botão direito do mouse sobre a pasta facebook nos arquivos do endereço local.

Nome



facebook



jogo1[Nova Versão]



jogo1[Nova Versão] Mobile

**Figura 199**

11. Selecione a opção “Upload”.

Nome	Tamanho	Ti
 facebook		
 jogo1(Nova	 Upload	
 jogo1(Nova	 Adicionar arquivos à fila	
 Nova pasta		 Entrar na pasta

## **Figura 200**

Aguarde os arquivos serem copiados para o servidor web. O próximo passo é alterar as configurações no Facebook.

## Atualizando as configurações do aplicativo no Facebook

Lembre que as configurações do aplicativo remetiam a uma conexão no servidor local. Agora iremos realizar as alterações nas configurações indicando o novo local do aplicativo no servidor web.

1. Abra o Google Chrome e digite a URL a seguir:

<https://developers.facebook.com/apps>

2. Clique no aplicativo Resgate.

3. Clique na opção “Configurações”.

**Ressgate**



**Panel de controle**



**Configurações**



**Status & Review**

## **Figura 201**

4. No item “App Domains” altere a URL para o endereço do domínio onde os arquivos estão hospedados.

Display Name

Regata

App Domains

www.muharraga.com

## **Figura 202**

5. No item “App on Facebook”, na opção “Canvas URL”, digite a URL da pasta onde os arquivos do aplicativo estão hospedados.

## App on Facebook

### Página Canvas

<https://apps.facebook.com/resgate>

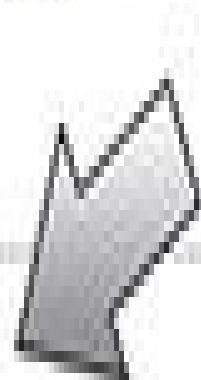


### Unity Integration

"Yes" to use the Facebook Unity SDK

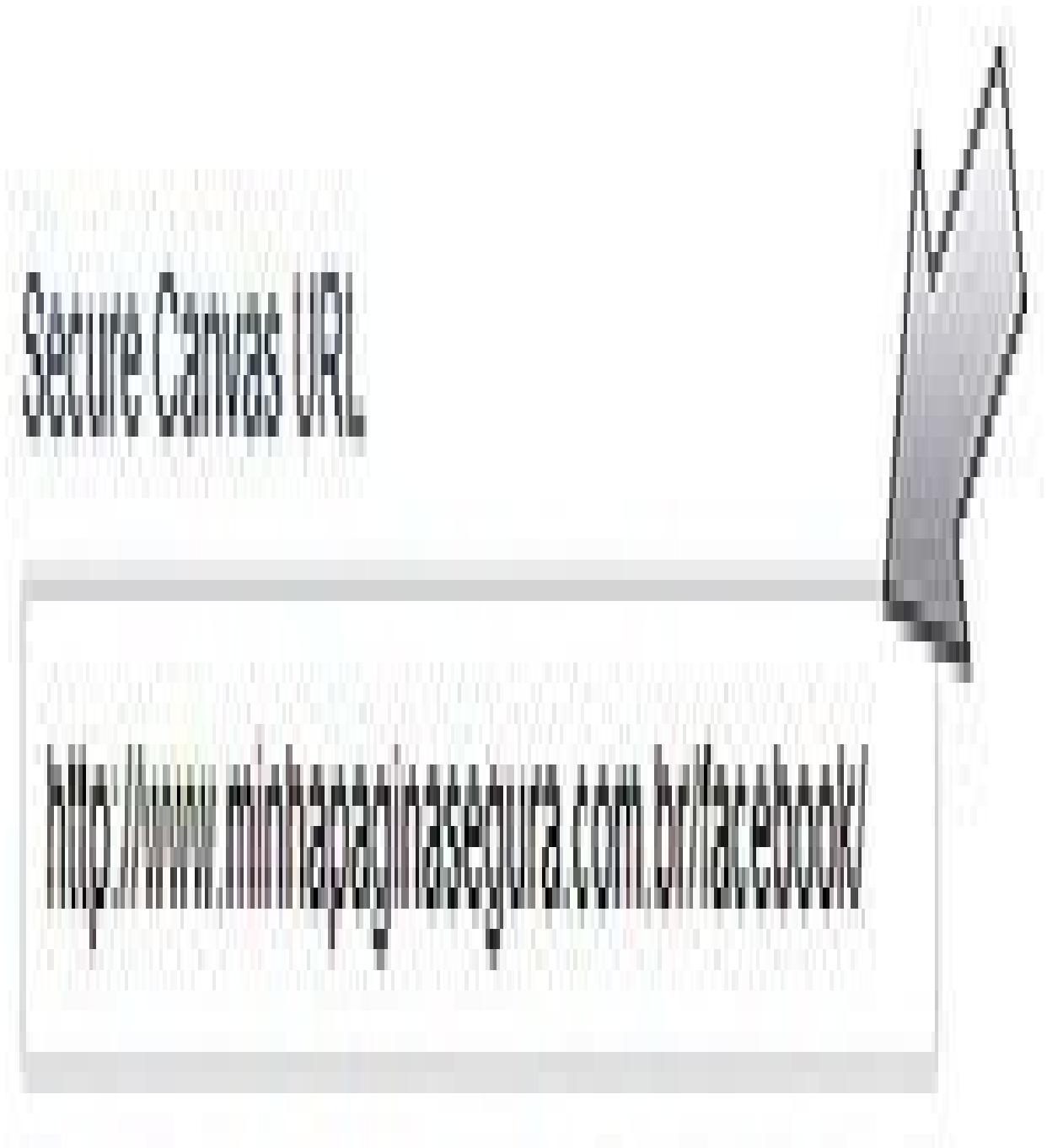
### URL

<http://www.minhapagina.com.br/facebook>



### **Figura 203**

6. No item “Secure Canvas URL”, digite o endereço da conexão segura do servidor web onde os arquivos estão hospedados.



## **Figura 204**

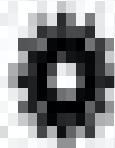
7. Clique no botão “Salvar alterações”.

O próximo passo é realizar as configurações finais, como indicar os ícones que serão utilizados, descrição do conteúdo do aplicativo, categoria etc. Todas essas configurações serão utilizadas para distribuir o aplicativo pelo Facebook appCenter, a loja de aplicativos do Facebook. Você pode ter acesso ao Facebook appCenter pela URL a seguir:

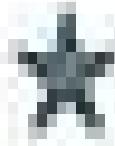
<https://www.facebook.com/appcenter/>

Todas as imagens utilizadas neste capítulo estão na pasta OutrosArquivos dos arquivos do livro, disponíveis em  
<http://www.denilsonbonatti.com.br/livros/download.php>.

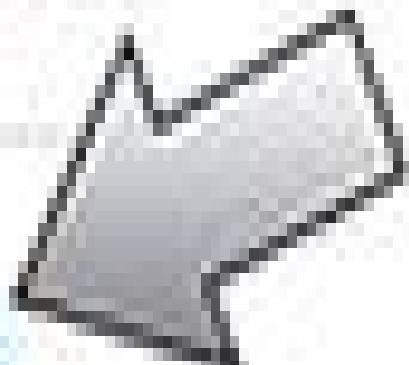
8. Clique na opção “App Details” (Figura 205).



## Configurações



## Status & Review



## App Details



## Roles

## **Figura 205**

As seguintes propriedades podem ser indicadas para o aplicativo:

**Idioma** – Indica o idioma em que o aplicativo foi desenvolvido.

**Slogan** – Slogan para o aplicativo com no máximo quarenta caracteres.

**Breve Descrição** – Breve descrição da função do aplicativo com no máximo 130 caracteres.

**Descrição Longa** – Descrição mais abrangente para o aplicativo com nome máximo de mil caracteres.

**Explanation for Permissions** – Aqui pode-se descrever quais informações do usuário o aplicativo irá obter (permissões) e o que o aplicativo irá realizar com essas permissões.

**Editor de publicações** – Nome da empresa ou proprietário do aplicativo.

**Categoria** – Aqui pode-se escolher de uma lista de categorias, como diversão, notícias, moda, educação etc., em qual se enquadra o aplicativo criado.

9. Como exemplo, os seguintes dados foram preenchidos:

**Idioma – Português (Brasil).**

**Slogan – Salve seus amigos do Facebook.**

**Breve Descrição – Resgate seus amigos do Facebook com esse jogo superdivertido.**

**Descrição Longa – Resgate seus amigos do Facebook com esse jogo superdivertido. Veja se você consegue superar os pontos de seus amigos!!**

**Explanation for Permissions – Para utilizar o jogo “Resgate” será necessário compartilhar conosco algumas informações pessoais e de seus amigos.**

**Editor de publicações – Denilson Bonatti.**

**Categoria – Diversão.**

10. Altere a opção “App on Facebook” para “Sim”, como indicado na Figura 206.



# App Center Listed Platforms



App on Facebook



App on Facebook

## **Figura 206**

O próximo passo é selecionar imagens e ícones que serão utilizados para a exibição do jogo no Facebook appCenter.

11. Dê um clique no item “Logotipo” (Figura 207).

Logo



1024 x 1024

## **Figura 207**

12. Selecione o arquivo logotipo.png da pasta OutrasImagens.
13. Dê um clique no item “Ícone (16x16)”.
14. Selecione o arquivo icone16x16.png da pasta OutrasImagens.
15. Dê um clique no item “Faixa da Web (155x100)”.
16. Selecione o arquivo banner155x100.png da pasta OutrasImagens.
17. Dê um clique no item “Imagen de Capa (800x150)”.
18. Selecione o arquivo banner800x150.png da pasta OutrasImagens.

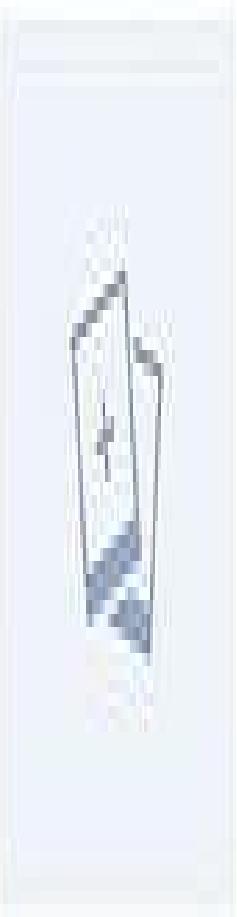
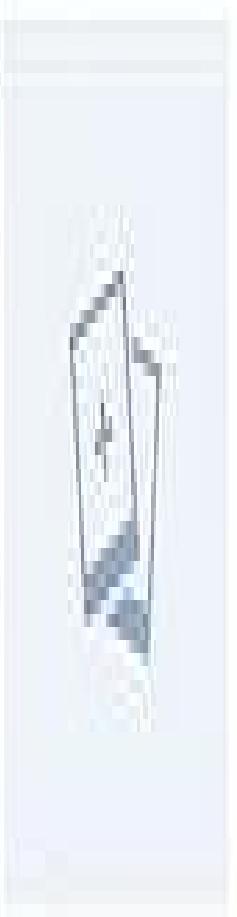
Não iremos utilizar as demais imagens opcionais.

O próximo passo é selecionar as imagens que serão utilizadas para demonstrar o jogo, chamadas de screenshots.

19. Iremos adicionar três screenshots. Selecione os arquivos screenshot1.png, screenshot2.png e screenshot3.png da pasta OutrasImagens, como indicado na Figura 208.

## Screenshots

App on Facebook



## **Figura 208**

20. Clique no botão “Salvar alterações”.

Vamos visualizar como o aplicativo será exibido no appCenter do Facebook.

21. Dê um clique no botão “Web Preview”, indicado pela Figura 209.



### **Figura 209**

Observe que serão exibidas as imagens e descrições cadastradas anteriormente (Figura 210).



RESGATE

## Resgate

[Ir para o aplicativo](#)

Aeronaves, Diversos

Terrenos

Informações

Compartilhar

Remover

Recuperar

Reibir uma fatura



## **Figura 210**

Com as configurações finalizadas, o próximo passo é deixar o jogo disponível para todos os usuários do Facebook através do seu link de acesso, neste exemplo:

<https://apps.facebook.com/resgate/>

22. Dê um clique no item “Status & Review” (Figura 211).

# Resgate



Painel de controle



Configurações



Status & Review

**Figura 211**

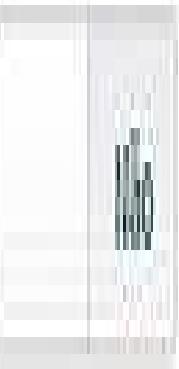
23. Altere a opção indicada pela Figura 212 para “Sim”.



Reggata

Reggata

Reggata



## **Figura 212**

24. Uma mensagem de configuração será exibida. Clique no botão “Confirmar”.

A partir de agora, o seu jogo estará disponível para todos os usuários do Facebook através da sua URL e estará disponível para buscas no Facebook appCenter depois que dez usuários se registrarem no jogo.

## Capítulo 27

### *Canvas game*

Inicialmente produzido pela Apple para o navegador Safari, o elemento <canvas> é utilizado para delimitar uma área do browser que irá receber imagens ou renderização de gráficos como: linhas, círculos, retângulos etc., que podem ser criados através do JavaScript.

Utilizando o elemento <canvas> juntamente com o JavaScript daremos início a um segundo jogo com as seguintes características:

#### **Nome do jogo – Pegue a Bandeira**

**Tema – Jogo para um jogador, que irá controlar um tanque com o objetivo de recolher um número maior de bandeiras do que o tanque controlado pelo computador.**

Ele poderá ser jogado online via browser e também em dispositivos móveis Android em forma de aplicativo.

Tanto a versão online como a versão em forma de aplicativo terá o seguinte layout (Figura 213):



Jugador 0



Computador 4



## **Figura 213**

**Controles – Para dificultar o jogo, o jogador apenas poderá girar o tanque em sentido horário. A cada clique ou toque na área do jogo o tanque irá girar 45° em sentido horário.**

**Fim de jogo – O jogo será finalizado quando o jogador ou o computador recolher cinco bandeiras.**

Começaremos o desenvolvimento deste jogo pelo seu layout.

1. Crie uma nova pasta com o nome de jogoCanvas.
2. Copie as pastas imgs, js e fontes da pasta jogo2 dos arquivos do curso para a pasta jogoCanvas.
3. Abra o Notepad ++.
4. Salve o arquivo dentro da pasta jogoCanvas com o nome de index.html.

O próximo passo é criar um código HTML5 básico para o jogo.

5. Digite o código a seguir.

```
<!doctype html>
```

```
<html>

<head>

<meta charset="utf-8">

<title>Pegue a Bandeira</title>

<link href="css/estilos.css" rel="stylesheet" type="text/css">

<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>

<script type="text/javascript" src="js/js.js"></script>

</head>

<body>

<h1>Pegue a bandeira!</h1>

<div id="fundo">

<div id="inicio" onClick="canvasApl()"><p class= "titulo">Bem-vindo ao
jogo!!</p><p class="paragrafo"> Clique ou toque aqui para iniciar</p>

</div>

<canvas id="canvasGame" width="500" height="500"></canvas>

</div>

</body>
```

```
</html>
```

6. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

Observe que no código anterior criamos o elemento canvas pela tag <canvas> com o nome de canvasGame definindo o seu tamanho em 500px de largura por 500px de altura.

Após definir o elemento canvas, dentro do código JavaScript, é possível desenhar retângulos, caminhos (paths) que incluem segmentos de linhas e arcos, posicionar arquivos de imagem sobre canvas etc. É possível também criar preenchimentos dentro de retângulos e caminhos. Nesse jogo em particular, iremos manipular o posicionamento e a exibição de imagens dentro do canvas criado.

Antes de criar o arquivo JavaScript, vamos criar o arquivo CSS para posicionar e formatar os elementos do jogo.

7. Crie um novo arquivo no Notepad++.

8. Crie uma nova pasta dentro da pasta jogoCanvas com o nome de css.

9. Salve o arquivo novo dentro da pasta css com o nome de estilos.css.

10. Digite o código a seguir.

```
@font-face {
```

```
font-family:Texto;
```

```
src:url(..../fontes/dirtyheadline.ttf);
```

```
}
```

```
body {
```

```
background-image:url(..../imgs/fundo.jpg);
```

```
}
```

```
h1 {
```

```
padding-top: 20px;
```

```
padding-bottom: 20px;
```

```
background-color:#666;
```

```
color:#FFF;
```

```
text-align:center;
```

```
font-family:Texto;
```

```
font-size: 50px;
```

}

```
#fundo {  
width:450px;  
height:450px;  
border-width:5px;  
border-style:solid;  
border-color:#FFF;  
margin:auto;  
position: relative;  
overflow: hidden;;  
z-index:1;
```

}

```
#inicio {  
width:350px;  
height:350px;  
background-color:#FFF;  
margin-left:auto;  
margin-right:auto;
```

margin-top:50px;

padding:5px;

z-index:1;

}

.titulo {

text-align:center;

font-family:Texto;

font-size: 50px;

color:#F00;

}

.paragrafo {

text-align:center;

font-family:Texto;

font-size: 30px;

color:#000;

}

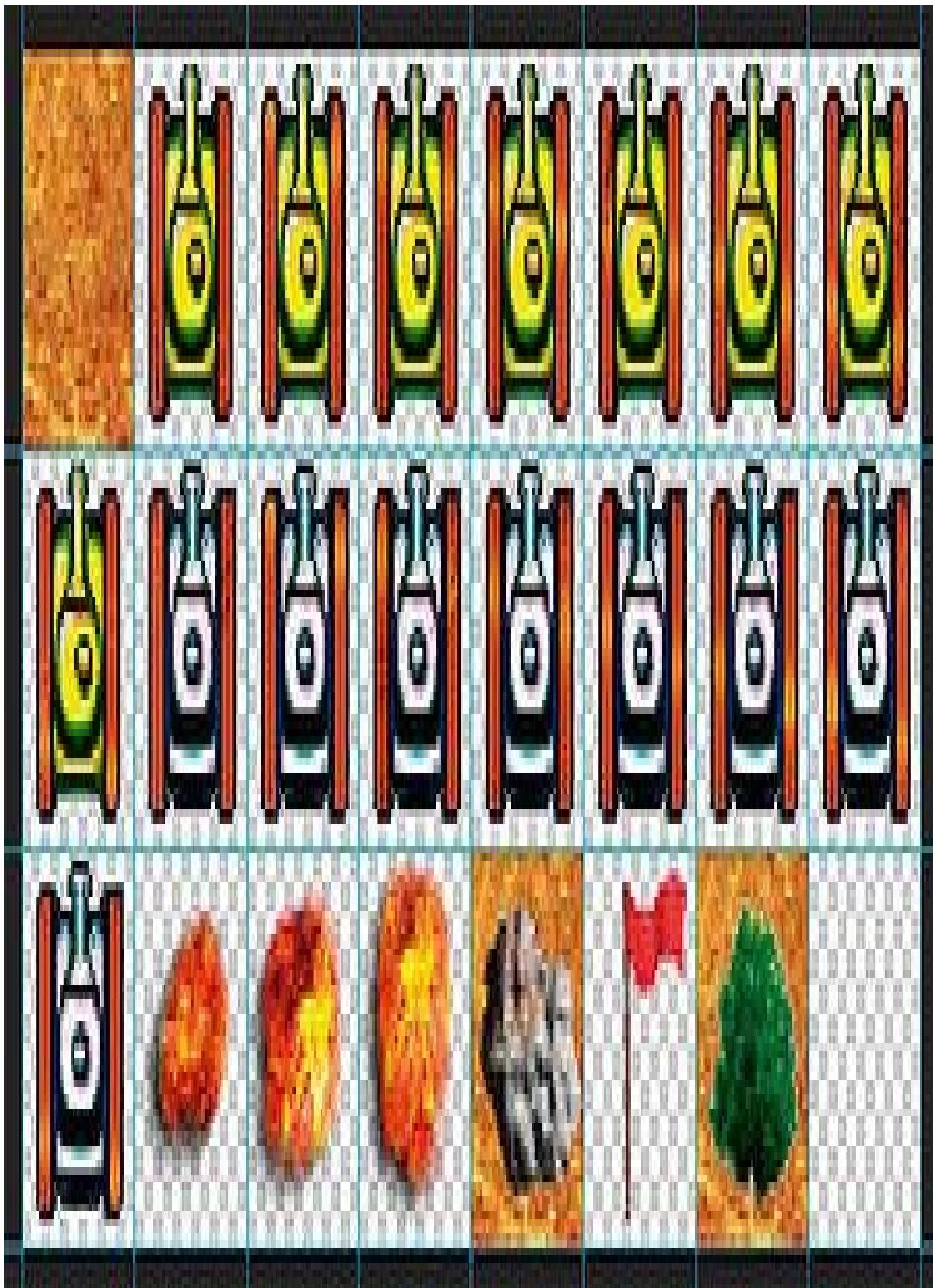
11. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.

O próximo passo é criar o arquivo JavaScript.

## Utilizando uma imagem mapeada

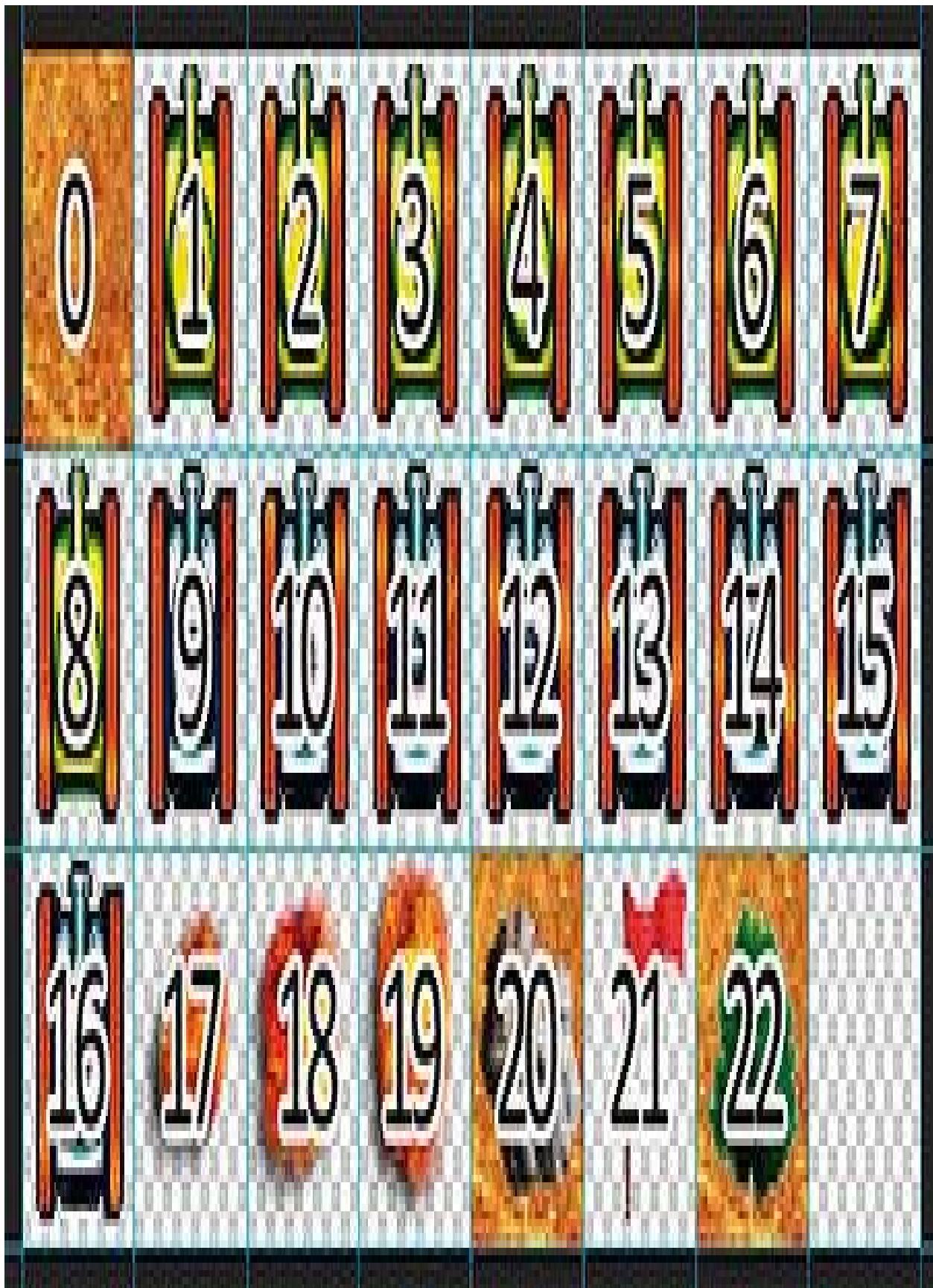
Com uma imagem mapeada, é relativamente simples criar uma animação baseada em células. Essa técnica envolve a troca rápida das imagens para simular uma animação.

Para utilizar uma imagem mapeada, antes é necessário criá-la. Na Figura 214 temos uma imagem criada com o Adobe Photoshop utilizando o tamanho de 32 pixels de largura por 32 pixels de altura para cada célula:



### **Figura 214**

Observe que a imagem possui oito células de largura por três células de altura. Dessa forma, podemos identificar cada um dos quadrantes da imagem (células) da seguinte forma:



## Figura 215

Vamos ao código que irá criar a imagem de fundo do jogo utilizando a imagem mapeada.

1. Crie um novo arquivo no Notepad++.
2. Salve o arquivo com o nome de js.js dentro da pasta js.
3. Digite o código a seguir:

```
function canvasApl(){  
    $("#inicio").hide();  
  
    var exibeCanvas = document.getElementById("canvasGame");  
  
    var context = exibeCanvas.getContext("2d");  
  
    var imagemMapeada=new Image();  
  
    imagemMapeada.addEventListener('load', gameLoop , false);  
  
    imagemMapeada.src="imgs/mapa.png";  
  
    //Variáveis de definição do mapa  
  
    var mapaLinhas = 15;  
  
    var mapaColunas = 15;  
  
    var Mapa = [
```

```
//Função GameLoop  
function gameLoop() {  
    setInterval(desenhaTela, 50 );
```

```
}
```

```
//Função responsável em desenhar o mapa no canvas
```

```
function desenhaTela() {
```

```
for (var linha=0;linha<mapaLinhas;linha++) {
```

```
for (var coluna=0;coluna<mapaColunas;coluna++){
```

```
var mapaId = Mapa[linha][coluna];
```

```
var sourceX = Math.floor(mapaId % 8) *32;
```

```
var sourceY = Math.floor(mapaId / 8) *32;
```

```
context.drawImage(imagemMapeada, sourceX,
```

```
sourceY,32,32,coluna*32,linha*32,32,32);
```

```
}
```

```
}
```

```
}
```

```
} //Fim da função canvasApl()
```

4. Pressione as teclas Ctrl + S para salvar o arquivo.



Observe que a função canvasApl será executada quando a div inicio for clicada.

```
<div id="inicio" onClick="canvasApl()"><p class="titulo">Bem-vindo ao  
jogo!!</p><p class="paragrafo">Clique ou toque aqui para iniciar</p>
```

```
</div>
```

Assim que a função for executada, a div inicio será ocultada pela função hide.

```
$("#inicio").hide();
```

Na função canvasApl definimos as variáveis que controlarão o elemento canvas. Criamos uma variável com o nome de exibeCanvas que receberá o elemento canvasGame criado no código HTML.

Na variável context indicamos que serão utilizados recursos de renderização em duas dimensões. Em seguida criamos uma variável com o nome de imagemMapeada que receberá uma imagem.

```
var exibeCanvas = document.getElementById("canvasGame");  
var context = exibeCanvas.getContext("2d");  
var imagemMapeada=new Image();
```

A seguir adicionamos um evento que executará a função gameLoop quando a imagem da variável imagemMapeada for carregada. Observe que indicamos que a imagem será carregada pela variável imagemMapeada através da propriedade src.

```
imagemMapeada.addEventListener('load', gameLoop , false);  
imagemMapeada.src="imgs/mapa.png";
```

A seguir temos as variáveis que definirão o tamanho e o design do mapa. Na variável mapaLinhas indicamos o tamanho da altura do mapa em células. Nesse exemplo, indicamos quinze células de altura. Na variável mapaColunas indicamos o tamanho da largura do mapa em células. Nesse exemplo, indicamos quinze células de largura.

Na variável Mapa indicamos quais células da imagem mapeadas aparecerão em cada uma das células do mapa.

```
//Variáveis de definição do mapa
```

```
var mapaLinhas = 15;  
var mapaColunas = 15;  
var Mapa = [  
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
, [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
, [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

Logo ap s, criamos a fun o gameLoop, que executar  a fun o desenhaTela a cada cinquenta milissegundos.

```
//Função GameLoop
```

```
setInterval(desenhaTela, 50 );
```

```
}
```

A função desenhaTela é a responsável por capturar cada uma das células da imagem mapa.png e exibi-las no canvas da forma como foram definidas na variável Mapa.

A função será executada dentro de dois loops: o primeiro define as linhas do mapa pela variável linha e o segundo define as colunas do mapa pela variável coluna. Os loops somente serão finalizados quando o número de linhas e colunas especificadas nas variáveis mapaLinhas e mapaColunas for finalizado.

Para encontrar o posicionamento no eixo X de cada uma das células definidas na variável Mapa foi criada uma variável com o nome de sourceX, que contém a fórmula (mapaId% número de colunas da imagem) \* largura da imagem.

O operador módulo (%) retorna o resto de uma divisão – por exemplo, a primeira célula está posicionada no valor 0 da variável mapaId, sendo assim o cálculo ficará: 0 % 8, onde o resultado é 0. Multiplicado por 32 tem-se o resultado 0, indicando o valor de 0 para a primeira célula no eixo X.

Na segunda célula o valor do mapaId é 1; sendo assim, o cálculo será 1 % 8, onde o resultado é 1. Multiplicado por 32 tem-se o resultado 32, indicando então o valor de 32 no eixo X para a segunda célula.

Na terceira célula o valor da variável mapaId será 2; sendo assim, o cálculo será  $2 \% 8$ , onde o resultado é 2. Multiplicado por 32 tem-se o resultado de 64, indicando o valor de 64 para a terceira célula no eixo X.

Para encontrar o valor da posição do eixo Y vamos dividir o valor do mapaId por 8 e multiplicar pelo tamanho da célula.

Para finalizar, foi utilizado o método `drawImage()` para exibir cada uma das células no canvas. Veja como ficou a função:

/Função responsável em desenhar o mapa no canvas

```
function desenhaTela() {  
  
    for (var linha=0;linha<mapaLinhas;linha++) {  
  
        for (var coluna=0;coluna<mapaColunas;coluna++) {  
  
            var mapaId = Mapa[linha][coluna];  
  
            var sourceX = Math.floor(mapaId % 8) *32;  
  
            var sourceY = Math.floor(mapaId / 8) *32;  
  
            context.drawImage(imagemMapeada, sourceX,  
                sourceY,32,32,coluna*32,linha*32,32,32);  
        }  
    }  
}
```

}

}

}

5. Execute o arquivo index.html no Google Chrome. Observe o resultado do mapa criado (Figura 216).



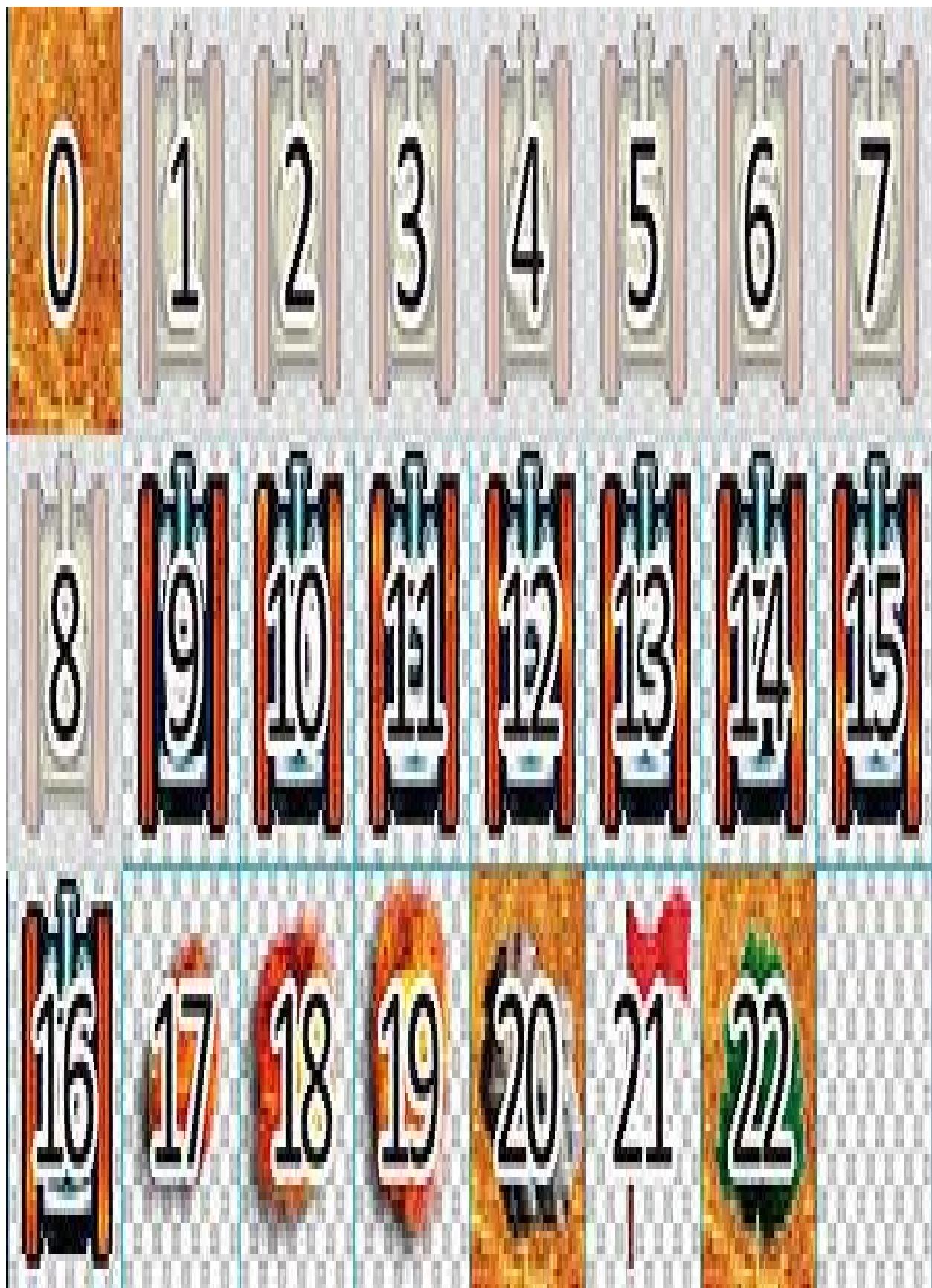
**Figura 216**

## Capítulo 28

### Criando uma animação com imagens mapeadas

Neste capítulo criaremos uma animação utilizando a imagem mapa.png. Com uma imagem mapeada, é relativamente simples criar uma animação baseada em células. Essa técnica envolve a troca rápida das imagens para simular uma animação.

Para criar a primeira animação utilizaremos as células 1, 2, 3, 4, 5, 6, 7 e 8. Veja as imagens que serão utilizadas na Figura 217.



## **Figura 217**

Vamos inicialmente criar as variáveis que serão utilizadas para gerar a animação.

1. No arquivo js.js, digite o código a seguir no local indicado pela Figura 218.

```
//Variáveis de definição do tanque1
var tanque1Frames=[1,2,3,4,5,6,7,8];
var tanque1Index=0;
var tanque1Rotation=90;
var tanque1X=Math.floor(Math.random() * 400);
var tanque1Y=Math.floor(Math.random() * 400);
```

```
, [0,0,0,0,0,0,0,0,0,0,0,0]
, [0,0,0,0,0,20,0,0,0,0,0,0]
];
```

```
//Variáveis de definição do tanque1
var tanque1Frames=[1,2,3,4,5,6,7,8];
var tanque1Index=0;
var tanque1Rotation=90;
var tanque1X=Math.floor(Math.random() * 400);
var tanque1Y=Math.floor(Math.random() * 400);
```

```
//Função GameLoop
function gameLoop() {
```

## **Figura 218**

Observe que criamos uma variável do tipo array com o nome de tanque1Frames, que recebeu a posição das células que serão utilizadas na animação.

Foi criada também uma variável com o nome de tanque1Index, que contém o valor de controle do índice da array. Esse índice será utilizado para identificar a célula que será exibida no decorrer da animação.

A seguir foi criada uma variável com o nome de tanque1Rotation, indicando a rotação inicial em graus para a exibição do tanque.

Para finalizar, foram criadas duas variáveis que receberão um valor aleatório entre 0 e 400; esses valores serão utilizados para posicionar o tanque1 em um local aleatório no mapa.

Agora vamos criar uma função com o nome de desenhaTanque1 para exibir o tanque1 no mapa.

2. Digite o código a seguir no local indicado na Figura 219.

```
//Função responsável em desenhar o tanque1 no mapa
```

```
function desenhaTanque1() {
```

```
angleInRadians =tanque1Rotation * Math.PI / 180;  
context.translate(tanque1X+16, tanque1Y+16);  
context.rotate(angleInRadians);  
var sourceX=Math.floor(tanque1Frames[tanque1Index] % 8) *32;  
var sourceY=Math.floor(tanque1Frames[tanque1Index] / 8) *32;  
context.drawImage(imagemMapeada, sourceX, sourceY,32,32,-16,-16,32,32);  
context.setTransform(1,0,0,1,0,0);  
  
tanque1Index++;  
if (tanque1Index ==tanque1Frames.length) {  
tanque1Index=0;  
}  
}
```

```
        var sourceY = Math.floor(mapaId / 8) *32;
        context.drawImage(imagemMapaEdu, sourceX,
        sourceY,32,32,coluna*32,linha*32,32,32);
    }
}
}

//Função responsável em desenhar o tanquei no mapa

function desenhaTanquei() {
    angleInRadians =tanqueiRotation * Math.PI / 180;
    context.translate(tanqueiX+16, tanqueiY+16);
    context.rotate(angleInRadians);
    var sourceX=Math.floor(tanqueiFrames[tanqueiIndex] % 8) *32;
    var sourceY=Math.floor(tanqueiFrames[tanqueiIndex] / 8) *32;
    context.drawImage(imagemMapaEdu, sourceX, sourceY,32,32,-16,-16,32,32);
    context.setTransform(1,0,0,1,0,0);

    tanqueiIndex++;
    if (tanqueiIndex ==tanqueiFrames.length) {
        tanqueiIndex=0;
    }
}

//?Fim da função canvasAp1()
```

**Figura 219**



Na função desenhaTanque1 inicialmente criamos uma variável com o nome de angleInRadians, que recebe como valor a fórmula PI/180 para converter o valor de graus para radiano.

```
angleInRadians =tanque1Rotation * Math.PI / 180;
```

Como o objeto será girado dentro do canvas, vamos inicialmente alterar o seu ponto de referência da matriz para o centro da imagem. Como cada uma das células mapeadas possui o tamanho de 32 pixels, movemos o ponto central na metade desse valor: 16 pixels.

```
context.translate(tanque1X+16, tanque1Y+16);  
context.rotate(angleInRadians);
```

A seguir foi criado o código para encontrar a posição X e Y de cada célula que será utilizada na animação. Observe que foi usado o mesmo método para o desenho do mapa.

```
var sourceX=Math.floor(tanque1Frames[tanque1Index] % 8) *32;  
var sourceY=Math.floor(tanque1Frames[tanque1Index] / 8) *32;
```

A seguir foi utilizado o método drawImage para exibir cada uma das células no canvas. A propriedade setTransform redefine a matriz de transformação HTML5 Canvas para seu estado padrão utilizando os valores 1, 0, 0, 1, 0, 0. Deve-se redefinir a matriz de transformação sempre que o canvas estiver dentro de um

loop.

```
context.drawImage(imagemMapeada, sourceX, sourceY, 32, 32, -16, -16, 32, 32);  
context.setTransform(1, 0, 0, 1, 0, 0);
```

Quando a variável tanque1Index possuir o mesmo valor da última célula da animação, a animação será reiniciada, alterando o valor da variável tanque1Index para 0.

```
tanque1Index++;
```

```
if (tanque1Index == tanque1Frames.length) {  
    tanque1Index = 0;  
}  
}
```

A função desenhaTanque1 deve ser chamada dentro do game loop do jogo para ser executada constantemente.

3. Digite o código a seguir no local indicado pela Figura 220.

```
setInterval(desenhaTanque1,50);
```

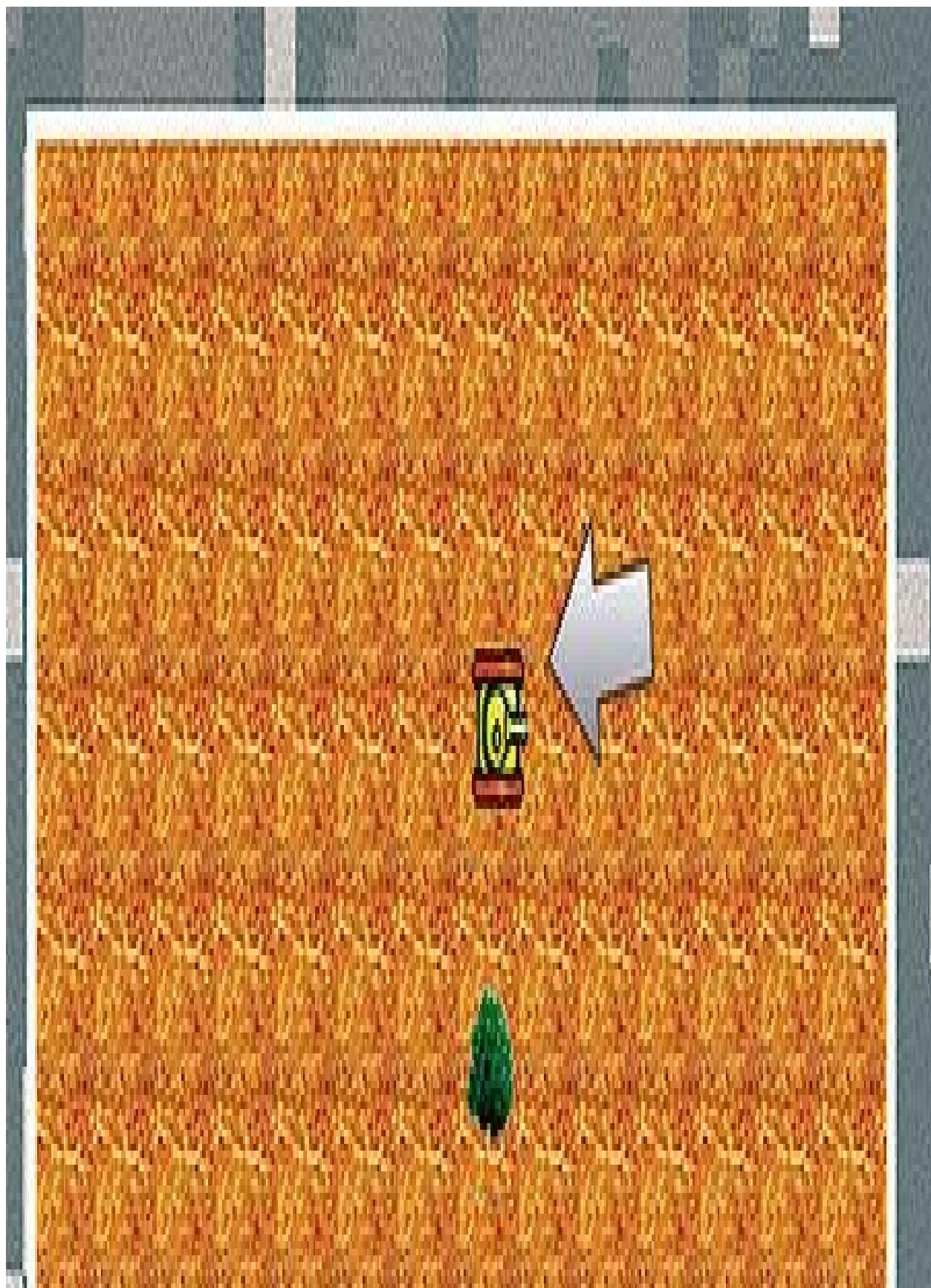
```
function gameLoop() {
```

```
    setInterval(desenhaTela, 50);
```

```
    setInterval(desenhaTela, 50);
```

### **Figura 220**

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html no Google Chrome e observe o resultado (Figura 221).



**Figura 221**

## Capítulo 29

### Criando a movimentação do tanque

A direção da movimentação do tanque1 será definida pelo jogador através do clique no mapa. Cada vez que o jogador clicar sobre o mapa, o tanque será girado em 45 graus no sentido horário. Vamos criar um evento que irá identificar o clique na área do mapa do jogo.

1. No arquivo js.js digite o código a seguir no local indicado na Figura 222.

```
exibeCanvas.addEventListener('click',eventoClick,false);
```

```
function canvasAp1() {  
    $("#t1inicio").hide();  
    var exibeCanvas = document.getElementById("canvasGame");  
    var context = exibeCanvas.getContext("2d");  
    var imagemMapa = new Image();  
    imagemMapa.addEventListener('load', gameLoop, false);  
    imagemMapa.src = "img/mapa.png";  
    exibeCanvas.addEventListener('click', eventoClick, false);  
}
```

**Figura 222**

## O que será executado na função eventoClick?

Criaremos uma variável com o nome de direcao, com o valor inicial de 1. Toda vez que a área do mapa for clicada, será somada uma unidade a essa variável, até ela receber o valor máximo de 4. Quando o mapa for clicado pela quinta vez, o seu valor será reiniciado para 1.

Assim, a variável irá conter a direção para a qual o tanque1 deverá se movimentar:

-

Valor da variável direcao	Direção da movimentação
1	Para a direita
2	Para baixo
3	Para a esquerda
4	Para cima

- 
- 1. Vamos inicialmente criar a função eventoClick. Digite o código a seguir no local indicado pela Figura 223.

```
//Função executada quando o mapa é clicado
```

```
function eventoClick() {  
    direcao++;  
  
    if (direcao==5) {  
        direcao=1;  
    }  
}
```

```
tanque1Index++;  
}  
if (tanque1Index == tanque1Frames.length) {  
    tanque1Index=0;  
}
```

```
//Função executada quando o mapa é clicado
}

function eventoClick() {
    direcao+=;
    if (direcao==5) {
        direcao=1;
    }
}

} //Fim da função canvasApl()
```

### **Figura 223**

2. O próximo passo é criar a variável direcao. Digite o código a seguir logo após a declaração da variável Mapa. Veja na Figura 224.

```
//Variáveis iniciais do jogo
```

```
var direcao=1;
```

```
    , [0,0,0,0,0,0,0,0,0,0,0,0]
    , [0,0,0,0,0,20,0,0,0,0,0,0]
];
}
```

```
// Variáveis iniciais do jogo
var direcao=1;
```

```
//Variáveis de definição do tanque1
var tanque1Frames=[1,2,3,4,5,6,7,8];
```

## Figura 224

Agora, dentro da função desenhaTanque1 vamos criar o código que movimentará o tanque1 de acordo com o valor da variável direcao. Além de indicar a direção para a qual o tanque1 deverá se movimentar, também devemos girar a imagem de acordo com a direção da movimentação. Iremos também limitar a movimentação do tanque1 dentro da área do canvas, evitando que o tanque1 saia da área de visualização do jogador.

3. Digite o código a seguir dentro da função desenhaTanque1 no local indicado pela Figura 225.

```
//Identifica a direção de movimentação
```

```
if (direcao==1) {
```

```
tanque1Rotation=90;
```

```
tanque1X=tanque1X+2;
```

```
}
```

```
if (direcao==2) {
```

```
tanque1Rotation=180;
```

```
tanque1Y=tanque1Y+2;  
}  
  
if (direcao==3) {
```

```
tanque1Rotation=270;  
  
tanque1X=tanque1X-2;  
}
```

```
if (direcao==4) {  
  
tanque1Rotation=0;  
  
tanque1Y=tanque1Y-2;  
}
```

```
// Limita movimentação  
  
if (tanque1X>=416) {
```

```
direcao=3;
```

```
}
```

```
if (tanque1X<=0) {
```

```
direcao=1;
```

```
}
```

```
if (tanque1Y<=0) {
```

```
direcao=2;
```

```
}
```

```
if (tanque1Y>=416) {
```

```
direcao=4;
```

```
}
```

```
function desenhaTanque1() {  
    angleInRadians =tanque1Rotation * Math.PI / 180;  
    context.translate(tanque1X+16, tanque1Y+16);  
    context.rotate(angleInRadians);  
    var sourceX=Math.floor(tanque1Frames[tanque1Index] % 8) *32;  
    var sourceY=Math.floor(tanque1Frames[tanque1Index] / 8) *32;  
    context.drawImage(imagemTanque1, sourceX, sourceY,32,32,-16,-16,32,32);  
    context.setTransform(1,0,0,1,0,0);  
  
    tanque1Index++;  
    if (tanque1Index ==tanque1Frames.length) {  
        tanque1Index=0;  
    }  
    //Identifica a direção de movimentação   
    if (direcao==1) {  
        tanque1Rotation=90;  
        tanque1X=tanque1X+2;  
    }  
}
```

**Figura 225**

Em caso de dúvida sobre a posição correta do código, consulte o arquivo Capítu

- 
4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
  5. Execute o arquivo index.html no Google Chrome. Observe que ao clicar sobre o mapa o tanque1 será girado e movimentado em sua nova posição.

## Exibindo o inimigo e a bandeira

Para desenhar o tanque2, que será controlado pelo computador, vamos iniciar criando as variáveis necessárias.

1. Digite o código a seguir logo após a declaração das variáveis do tanque1. Veja na Figura 226.

```
//Variáveis de definição do tanque2 (inimigo)  
var tanque2Frames=[9,10,11,12,13,14,15,16];  
var tanque2Index=0;  
var tanque2Rotation=90;  
var tanque2X=Math.floor(Math.random() * 400);  
var tanque2Y=Math.floor(Math.random() * 400);
```

```
//Variáveis de definição da bandeira  
var bandeira = [0,21,21,21,21,0];  
var bandeiraIndex=0;  
var bandeiraX=Math.floor(Math.random() * 400);  
var bandeiraY=Math.floor(Math.random() * 400);
```



```
//Variáveis de definição do tanque1
var tanque1Frames=[1,2,3,4,5,6,7,8];
var tanque1Index=0;
var tanque1Rotation=90;
var tanque1X=Math.floor(Math.random() * 400);
var tanque1Y=Math.floor(Math.random() * 400);
```

```
//Variáveis de definição do tanque2 (inimigo)
var tanque2Frames=[9,10,11,12,13,14,15,16];
var tanque2Index=0;
var tanque2Rotation=90;
var tanque2X=Math.floor(Math.random() * 400);
var tanque2Y=Math.floor(Math.random() * 400);
```

```
//Variáveis de definição da bandeira
var bandeira = [0,21,21,21,21,0];
var bandeiraIndex=0;
var bandeiraX=Math.floor(Math.random() * 400);
var bandeiraY=Math.floor(Math.random() * 400);
```

## **Figura 226**

O próximo passo é criar as funções que exibirão o inimigo e a bandeira no canvas.

2. Digite o código a seguir no local indicado na Figura 227.

```
//Função responsável por desenhar o tanque2 (inimigo) no mapa
```

```
function desenhaInimigo() {
```

```
    var angleInRadians2 = tanque2Rotation * Math.PI / 180;
```

```
    context.translate(tanque2X+16, tanque2Y+16);
```

```
    context.rotate(angleInRadians2);
```

```
    var InimigoX=Math.floor(tanque2Frames[tanque2Index] % 8) *32;
```

```
    var InimigoY=Math.floor(tanque2Frames[tanque2Index] / 8) *32;
```

```
context.drawImage(imagemMapeada, InimigoX, InimigoY,32,32,-16,-16,32,32);
```

```
context.setTransform(1,0,0,1,0,0);
```

```
tanque2Index++;
```

```
if (tanque2Index ==tanque2Frames.length) {
```

```
tanque2Index=0;
```

```
}
```

```
}
```

```
//Função responsável por desenhar a bandeira no mapa
```

```
function desenhaBandeira() {
```

```
context.translate(bandeiraX+16, bandeiraY+16);
```

```
var BandeiraX=Math.floor(bandeira[bandeiraIndex] % 8) *32;
```

```
var BandeiraY=Math.floor(bandeira[bandeiraIndex] / 8) *32;
```

```
context.drawImage(imagemMapeada, BandeiraX,  
BandeiraY,32,32,-16,-16,32,32);
```

```
context.setTransform(1,0,0,1,0,0);
```

```
bandeiraIndex++;
```

```
if (bandeiraIndex ==bandeira.length) {
```

```
bandeiraIndex=0;
```

```
}
```

```
}
```

```
//Função executada quando o mapa é clicado
```

```
function eventoClick() {  
    direcao++;  
  
    if (direcao==5) {  
        direcao=1;  
    }  
}
```



```
//Função responsável em desenhar o tanque2 (inimigo) no mapa
```

```
function desenhalInimigo() {  
  
    var angleInRadians2 =tanque2Rotation * Math.PI / 180;  
    context.translate(tanque2X+16, tanque2Y+16);
```

**Figura 227**

Em caso de dúvida sobre a posição correta do código, consulte o arquivo Capítu

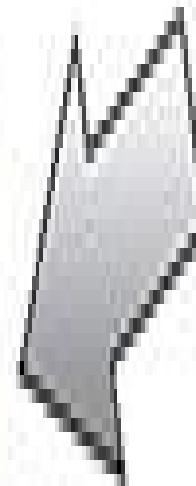
Para finalizar, as funções desenhaInimigo e desenhaBandeira devem ser chamadas no game loop do jogo.

3. Digite o código a seguir no local indicado na Figura 228.

```
setInterval(desenhaInimigo,50);  
setInterval(desenhaBandeira,50);
```

```
//Função GameLoop

function gameLoop() {
    setInterval(desenhaTela, 50);
    setInterval(desenhaCanhao, 50);
    setInterval(desenhaInimigo, 50);
    setInterval(desenhaBandeira, 50);
}
```



### **Figura 228**

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
5. Execute o arquivo index.html no Google Chrome. Observe que o tanque2 e a bandeira foram exibidos no canvas (Figura 229).



**Figura 229**

## Capítulo 30

### Inteligência artificial

Inteligência artificial, ou simplesmente IA, é o que controla e orienta oponentes computadorizados para proporcionar desafios ao jogador.

Neste capítulo vamos criar a inteligência artificial que controlará o tanque inimigo. Será uma inteligência artificial rudimentar, que apenas fará com que o tanque inimigo chegue o mais rápido possível onde está a bandeira. Criaremos esse comportamento fazendo um comparativo dos valores das posições X e Y do tanque2 com os valores da posição X e Y da bandeira, fazendo com que o tanque2 se movimente até o local da bandeira.

Criaremos esse comportamento dentro da função desenhaInimigo.

1. Digite o código a seguir no local indicado pela Figura 230.

```
//Inteligência Artificial
```

```
if (tanque2X>bandeiraX) {
```

```
tanque2X-=1;  
tanque2Rotation=270;  
}  
  
  
if (tanque2X<bandeiraX) {  
tanque2X+=1;  
tanque2Rotation=90;  
}  
  
  
if (tanque2Y>bandeiraY) {  
tanque2Y-=1;  
tanque2Rotation=0;  
}  
  
  
if (tanque2Y<bandeiraY) {  
tanque2Y+=1;  
tanque2Rotation=180;  
}  
}
```

```
tanque2Index++;

    if (tanque2Index == tanque2Frames.length) {
        tanque2Index=0;
    }

    //Inteligência Artificial

    if (tanque2X>bandeiraX) {
        tanque2X-=1;
        tanque2Rotation=270;
    }
    if (tanque2X<bandeiraX) {
        tanque2X+=1;
        tanque2Rotation=90;
    }
    if (tanque2Y>bandeiraY) {
        tanque2Y-=1;
        tanque2Rotation=0;
    }
    if (tanque2Y<bandeiraY) {
        tanque2Y+=1;
        tanque2Rotation=180;
    }
}
```

**Figura 230**



Se o valor da posição X do tanque2 for maior que a posição X da bandeira, o valor de X do tanque2 será subtraído, fazendo com que fique na mesma posição X da bandeira.

```
if (tanque2X>bandeiraX) {  
    tanque2X-=1;  
    tanque2Rotation=270;  
}  
}
```

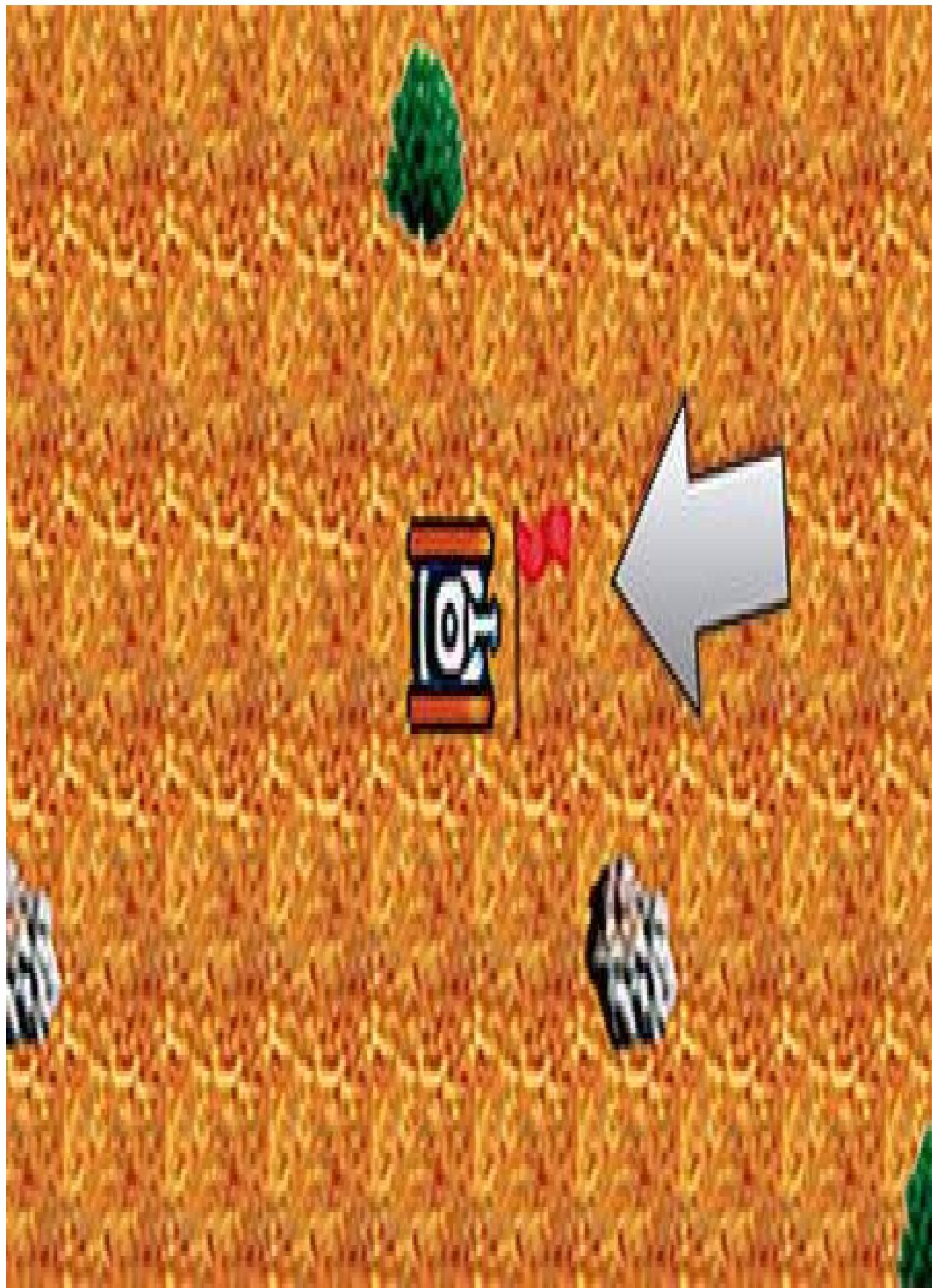
```
if (tanque2X<bandeiraX) {  
    tanque2X+=1;  
    tanque2Rotation=90;  
}  
}
```

Caso o valor de Y do tanque2 seja menor que o valor de Y da bandeira, o valor de Y do tanque2 será somado, de forma que ele fique na mesma posição Y da bandeira, e assim por diante.

```
if (tanque2Y>bandeiraY) {  
    tanque2Y-=1;
```

```
tanque2Rotation=0;  
  
}  
  
if (tanque2Y<bandeiraY) {  
tanque2Y+=1;  
tanque2Rotation=180;  
  
}
```

2. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.
3. Execute o arquivo index.html e veja o resultado. Observe que o tanque inimigo irá se movimentar automaticamente até a posição da bandeira:



## **Figura 231**

O próximo passo é detectar a colisão do tanque inimigo com a bandeira, fazendo com que a bandeira seja reposicionada em um novo local.

Vamos verificar esse comportamento dentro da função desenhaBandeira. Como o tanque inimigo é movimentado uma unidade por vez dentro dos eixos X e Y, vamos comparar se o valor de X e Y do tanque inimigo é o mesmo valor de X e Y da bandeira, detectando assim a colisão.

4. Digite o código a seguir na função desenhaBandeira no local indicado pela Figura 232.

```
//Detecta colisão da bandeira com o inimigo
```

```
if ((bandeiraX==tanque2X) && (bandeiraY==tanque2Y)) {  
    bandeiraX=Math.floor(Math.random() * 400);  
    bandeiraY=Math.floor(Math.random() * 400);  
}
```

```
bandeiraIndex++;  
if (bandeiraIndex == bandeira.length) {  
    bandeiraIndex=0;  
}  
}
```

```
//Detecta colisão da bandeira com o inimigo
```

```
if ((bandeiraX==tanque2X) && (bandeiraY==tanque2Y)) {  
bandeiraX=Math.floor(Math.random() * 400);  
bandeiraY=Math.floor(Math.random() * 400);  
}  
}
```

```
 } //Fim da função canvasApi()
```

## **Figura 232**

5. Pressione Ctrl + S para salvar as alterações no arquivo.
6. Execute o arquivo index.html e veja o resultado. Observe que a bandeira será exibida em uma nova posição quando o tanque inimigo entrar em colisão com ela.

Esse sistema não poderá ser utilizado para detectar a colisão do tanque1 controlado pelo jogador com a bandeira, pois o tanque1 não se movimenta uma unidade por vez nos eixos X e Y, e sim duas unidades; dessa forma, os valores de X e Y do tanque1 nunca serão iguais aos valores X e Y da bandeira.

Para detectar a colisão faremos um comparativo de X e Y do tanque1 somando 16 unidades com os valores de X e Y da bandeira. Estamos utilizando o valor de 16 para que a colisão seja identificada quando o tanque1 estiver bem próximo da bandeira. Para controlar a área de colisão basta aumentar ou diminuir esse valor de referência.

7. Digite o código a seguir na função desenhaBandeira no local indicado na Figura 233.

```
//Detecta colisão da bandeira com o tanque1
```

```
if (((bandeiraX + 16) > tanque1X && bandeiraX < (tanque1X + 16)) &&  
((bandeiraY + 16) > tanque1Y && bandeiraY < (tanque1Y + 16))) {
```

```
bandeiraX=Math.floor(Math.random() * 400);
```

```
bandeiraY=Math.floor(Math.random() * 400);
```

```
}
```

```
//Detecta colisão da bandeira com o inimigo
if ((bandeiraX==tanque2X) && (bandeiraY==tanque2Y)) {
bandeiraX=Math.floor(Math.random() * 400);
bandeiraY=Math.floor(Math.random() * 400);
}

//Detecta colisão da bandeira com o tanque1
if (((bandeiraX + 16) > tanque1X && bandeiraX < (tanque1X + 16)) &&
((bandeiraY + 16) > tanque1Y && bandeiraY < (tanque1Y + 16))) {
bandeiraX=Math.floor(Math.random() * 400);
bandeiraY=Math.floor(Math.random() * 400);
}

}

//Fim da função canvasAp1()
```

### **Figura 233**

8. Pressione Ctrl + S para salvar as alterações no arquivo
9. Execute o arquivo index.html e veja o resultado. Observe que a bandeira será exibida em uma nova posição quando o tanque1 entrar em colisão com ela.

## Capítulo 31

### Mensagens de início e fim de jogo

Para exibir a pontuação no canvas vamos criar uma nova função com o nome de desenhaTexto.

1. Digite o código a seguir no local indicado na Figura 234.

```
//Desenha o texto no canvas
```

```
function desenhaTexto() {  
  
    context.fillStyle = "rgb(250, 250, 250)";  
  
    context.font = "16px BRAESIDE";  
  
    context.textAlign = "left";  
  
    context.textBaseline = "top";  
  
    context.fillText("Jogador: " + pontosjog, 5, 5);  
  
    context.fillText("Computador: " + pontoscomp, 330, 5);
```

}

```
bandeiraX=Math.floor(Math.random() * 400);
bandeiraY=Math.floor(Math.random() * 400);

}

}

//Desenha o texto no canvas

function desenhaTexto() {

    context.fillStyle = "rgb(250, 250, 250)";
    context.font = "16px BRAESIDE";
    context.textAlign = "left";
    context.textBaseline = "top";
    context.fillText("Jogador: " + pontosjog, 5, 5);
    context.fillText("Computador: " + pontoscomp, 330, 5);

}

//Fim da função canvasApl()
```

### **Figura 234**

2. O próximo passo é criar as variáveis pontosjog e pontoscomp. Digite o código a seguir no local indicado na Figura 235.

```
var pontosjog=0;
```

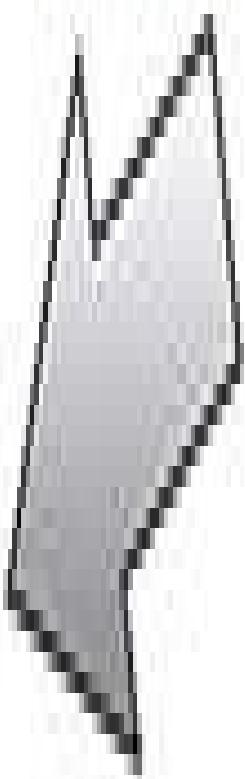
```
var pontoscomp=0;
```

// Variáveis iniciais do jogo

var descaida;

var pontologo;

var contocom;



### **Figura 235**

Agora vamos somar a variável pontosjog toda vez que o tanque1 entrar em colisão com a bandeira e somar a variável pontoscomp toda vez que o tanque inimigo entrar em colisão com a bandeira.

3. Utilize os comandos pontosjog++ e pontoscomp++ nos locais indicados na Figura 236, função desenhaBandeira.

```
//Detecta colisão da bandeira com o inimigo  
  
if ((bandeiraX==tanque2X) && (bandeiraY==tanque2Y)) {  
  
bandeiraX=Math.floor(Math.random() * 400);  
bandeiraY=Math.floor(Math.random() * 400);  
  
pontoscomp++;  
}  
  
}
```

```
//Detecta colisão da bandeira com o tanque1  
  
if (((bandeiraX + 16) > tanque1X && bandeiraX < (tanque1X + 16)) &&  
( (bandeiraY + 16) > tanque1Y && bandeiraY < (tanque1Y + 16))) {  
  
bandeiraX=Math.floor(Math.random() * 400);  
bandeiraY=Math.floor(Math.random() * 400);  
  
pontosjog++;  
}  
  
}
```

## **Figura 236**

Para finalizar o jogo criaremos a função gameOver. Ela irá detectar o fim do jogo quando o jogador1 ou o computador atingir cinco pontos. Quem atingir cinco pontos primeiro será o vencedor.

4. Digite o código a seguir no local indicado na Figura 237.

```
//Função GameOver
function gameOver() {
    if (pontosjog==5) {
        alert ("Você Ganhou");
        pontosjog=0;
        pontoscomp=0;
        window.location.reload();
    }
}
```

}

if (pontoscomp==3) {

    alert ("Você Perdeu");

    pontosjog=0;

    pontoscomp=0;

    window.location.reload();

}

}

```
//Função GameOver

function gameOver() {

    if (pontosjog==5) {

        alert ("Você Ganhou");
        pontosjog=0;
        pontoscomp=0;

        window.location.reload();
    }

    if (pontoscomp==5) {

        alert ("Você Perdeu");
        pontosjog=0;
        pontoscomp=0;
        window.location.reload();
    }
}

//Fim da função canvasApl()
```

## Figura 237

5. O próximo passo é chamar as funções gameOver e desenhoTexto no game loop do jogo. Digite o código indicado em negrito na função

```
//Função GameLoop

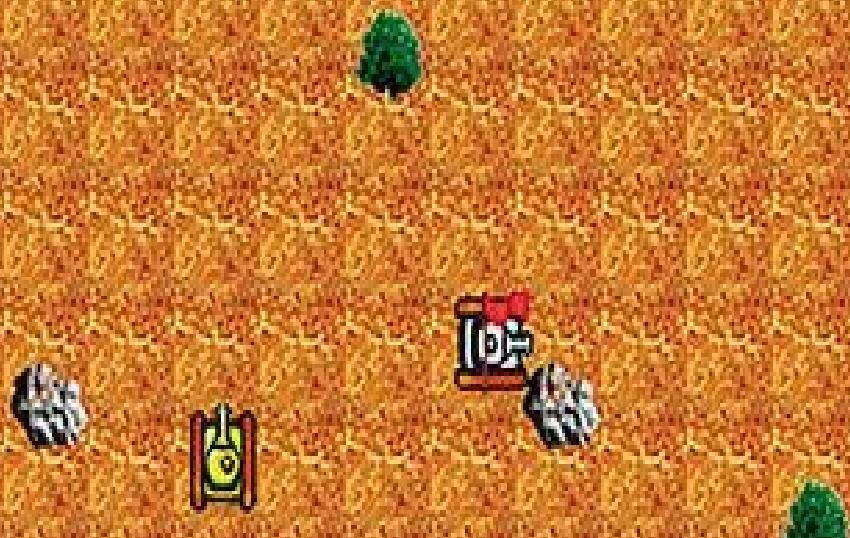
function gameLoop() {
    setInterval(desenhaTela,50);
    setInterval(desenhaTanque1,50);
    setInterval(desenhaInimigo,50);
    setInterval(desenhaBandeira,50);
    setInterval(desenhaTexto,50);
    setInterval(gameOver,50);
}
```

6. Pressione Ctrl + S para salvar as alterações no arquivo.

7. Execute o arquivo index.html e veja o resultado. Observe que a pontuação será exibida e a mensagem em forma de janela aparecerá quando o jogo chegar ao fim.

Jogador 2

Computador 5



Alerta JavaScript

Você Perdeu

X

OK

**Figura 238**

## Capítulo 32

### Utilizando o PhoneGap

O jogo no formato atual, caso esteja hospedado em um servidor web, pode ser executado em qualquer dispositivo móvel via browser, desde que haja resolução suficiente para exibi-lo. O que iremos realizar neste capítulo é converter o jogo do formato HTML/JS/CSS para um aplicativo nativo Android. Para isso utilizaremos o PhoneGap.

O PhoneGap é um framework de desenvolvimento móvel produzido pela empresa Nitobi, comprada recentemente pela Adobe Systems. O PhoneGap converte arquivos criados em JavaScript, HTML5 e CSS3 em aplicativos nativos para iPad (iOS), Android e Windows 8. Assim, um mesmo projeto pode ser distribuído para diversos sistemas operacionais usando apenas uma fonte de código.

Antes de convertermos o jogo em aplicativo móvel, devemos criar um arquivo XML que terá todas as informações, como versão, ícone a ser utilizado, propriedade do aplicativo etc.

1. Crie um novo arquivo no Notepad++.
2. Salve-o na pasta jogoCanvas com o nome de config.xml (na mesma pasta do arquivo index.html).

Criaremos o código XML padrão para aplicativos de dispositivos Android.

3. Digite o código a seguir.

```
<?xml version="1.0" encoding="UTF-8" ?>
<widget xmlns = "http://www.w3.org/ns/widgets"
  xmlns:gap = "http://phonegap.com/ns/1.0"
  id = "com.bandeira.mobile"
  versionCode="1"
  version = "1.0.0">

<name>Pegue a Bandeira</name>
<content src="index.html" />
<description>
  Tente recolher mais bandeiras que o seu inimigo!!
</description>

<author href="http://www.denilsonbonatti.com.br" email=
  "contato@denilsonbonatti.com.br">
</author>
```

```
<preference name="fullscreen" value="true" />  
<preference name="orientation" value="portrait" />  
<icon src="imgs/icon.png" />  
  
</widget>
```

4. Pressione as teclas Ctrl + S para salvar as alterações no arquivo.



Inicialmente, é indicado o cabeçalho padrão para o arquivo XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Indicamos pela tag <widget> as informações de identificação do aplicativo.

```
<widget xmlns = "http://www.w3.org/ns/widgets"  
xmlns:gap = "http://phonegap.com/ns/1.0"
```

Após isso, é indicado o id do aplicativo – neste exemplo, indicamos o caminho “com.bandeira.mobile”.

```
id = "com.bandeira.mobile"
```

A seguir, indicamos as versões do código e do aplicativo.

```
versionCode="1"  
version = "1.0.0"
```

Prosseguindo com o código, temos o nome do aplicativo indicado pela tag <name>, a descrição do aplicativo pela tag <description> e as informações do

autor indicadas pela tag <author>.

```
<name>Pegue a Bandeira</name>  
<content src="index.html" />  
<description>  
Tente recolher mais bandeiras que o seu inimigo!!  
</description>  
<author href="http://www.denilsonbonatti.com.br"  
email="contato@denilsonbonatti.com.br">  
</author>
```

Logo após, indicamos a forma como o aplicativo será executado no dispositivo móvel. No código, indicamos que o aplicativo será executado em tela cheia (fullscreen) e na posição vertical (portrait).

```
<preference name="fullscreen" value="true" />  
<preference name="orientation" value="portrait" />
```

Para finalizar, indicamos a imagem que será utilizada como ícone do aplicativo.

```
<icon src="imgs/icon.png" />
```

Para converter os arquivos do jogo em um aplicativo, devemos compactar todos os arquivos utilizados no jogo em um arquivo do tipo ZIP.

5. Selecione todos os arquivos da pasta jogoCanvas e compacte-os em um arquivo do tipo ZIP com o nome de jogoCanvas.zip.

■

Não utilize o formato .RAR para compactar os arquivos, utilize o formato .ZIP.

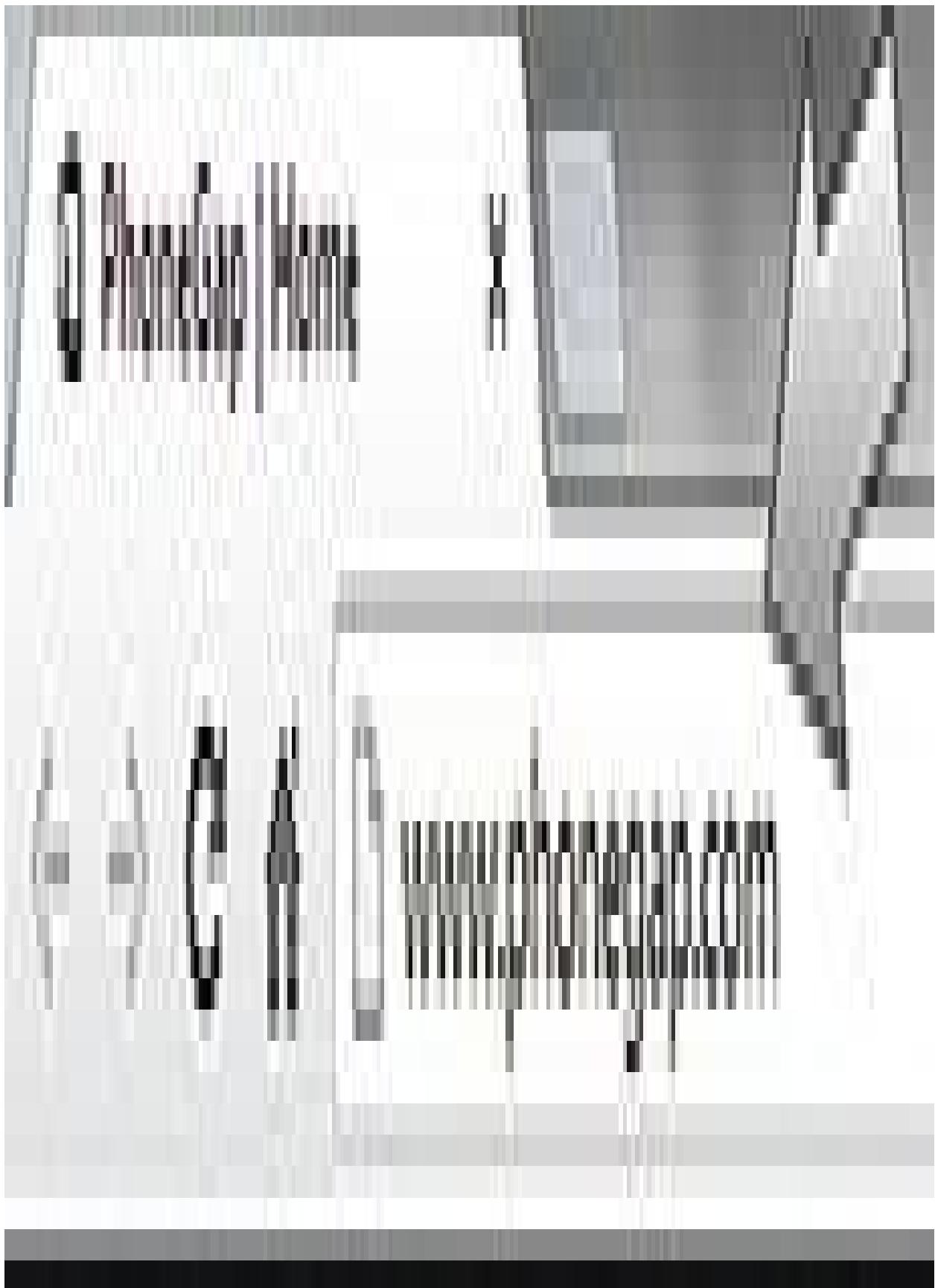




### **Figura 239**

Agora vamos utilizar o PhoneGap para converter o arquivo zip em um aplicativo.

6. Abra o Google Chrome e digite a URL [www.phonegap.com](http://www.phonegap.com), como indicado na Figura 240.



### **Figura 240**

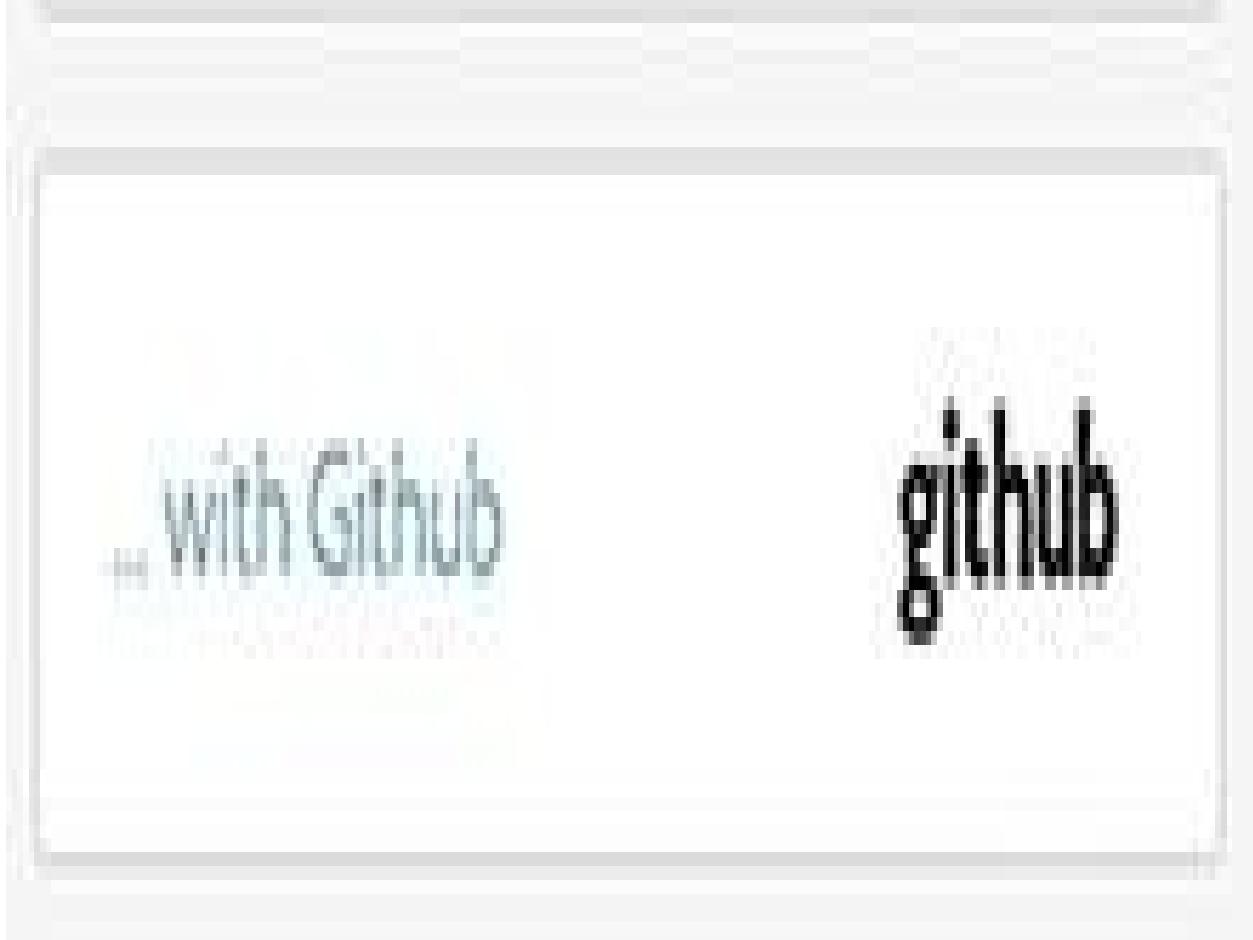
7. Quando a página for exibida, dê um clique no link “Try it now!”, como indicado na Figura 241.



### **Figura 241**

Para utilizar o PhoneGap você precisa de uma conta chamada de Adobe ID. Caso você não tenha essa identificação, será exibido um formulário de cadastro, onde você deverá criar um nome de usuário e uma senha para ter acesso ao PhoneGap e a outros produtos da Adobe.

8. Dê um clique na opção “with Adobe ID” (Figura 242).



### **Figura 242**

9. Acesse a página do PhoneGap utilizando o Adobe ID.
10. Quando a página for exibida, clique no botão “Upload a .zip file” (Figura 243).

Upload a zip file

### **Figura 243**

11. Selecione o arquivo jogoCanvas.zip e aguarde o upload do arquivo.

Observe que as informações indicadas no arquivo config.xml serão exibidas na página do PhoneGap.



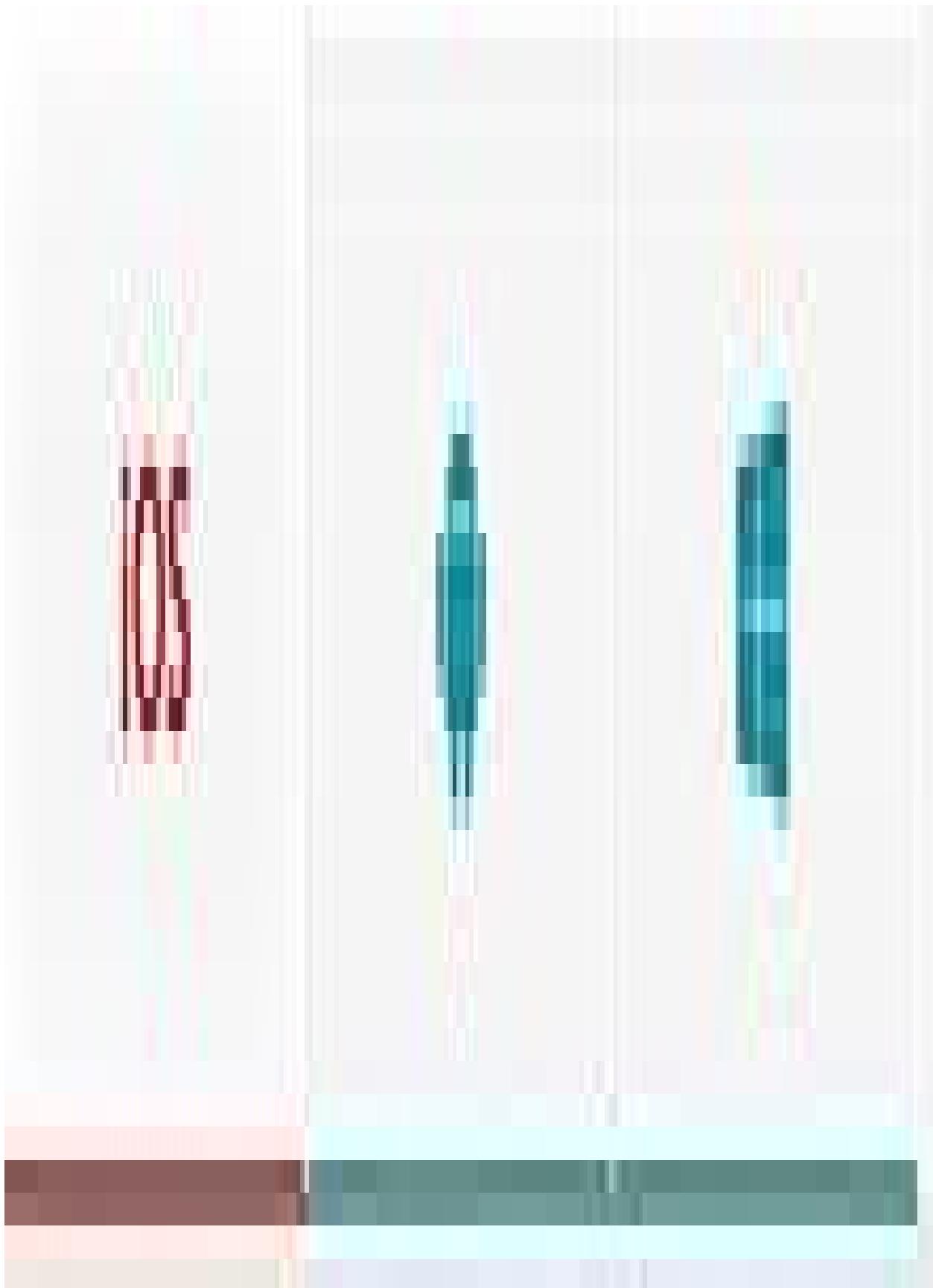
### **Figura 244**

12. Clique no botão “Ready to build” (Figura 245).



### **Figura 245**

Você vai notar que o único aplicativo que não será gerado será para iOS, pois é necessário um registro de identificação de desenvolvedor fornecido pela Apple para criar aplicativos para esse sistema operacional (Figura 246).



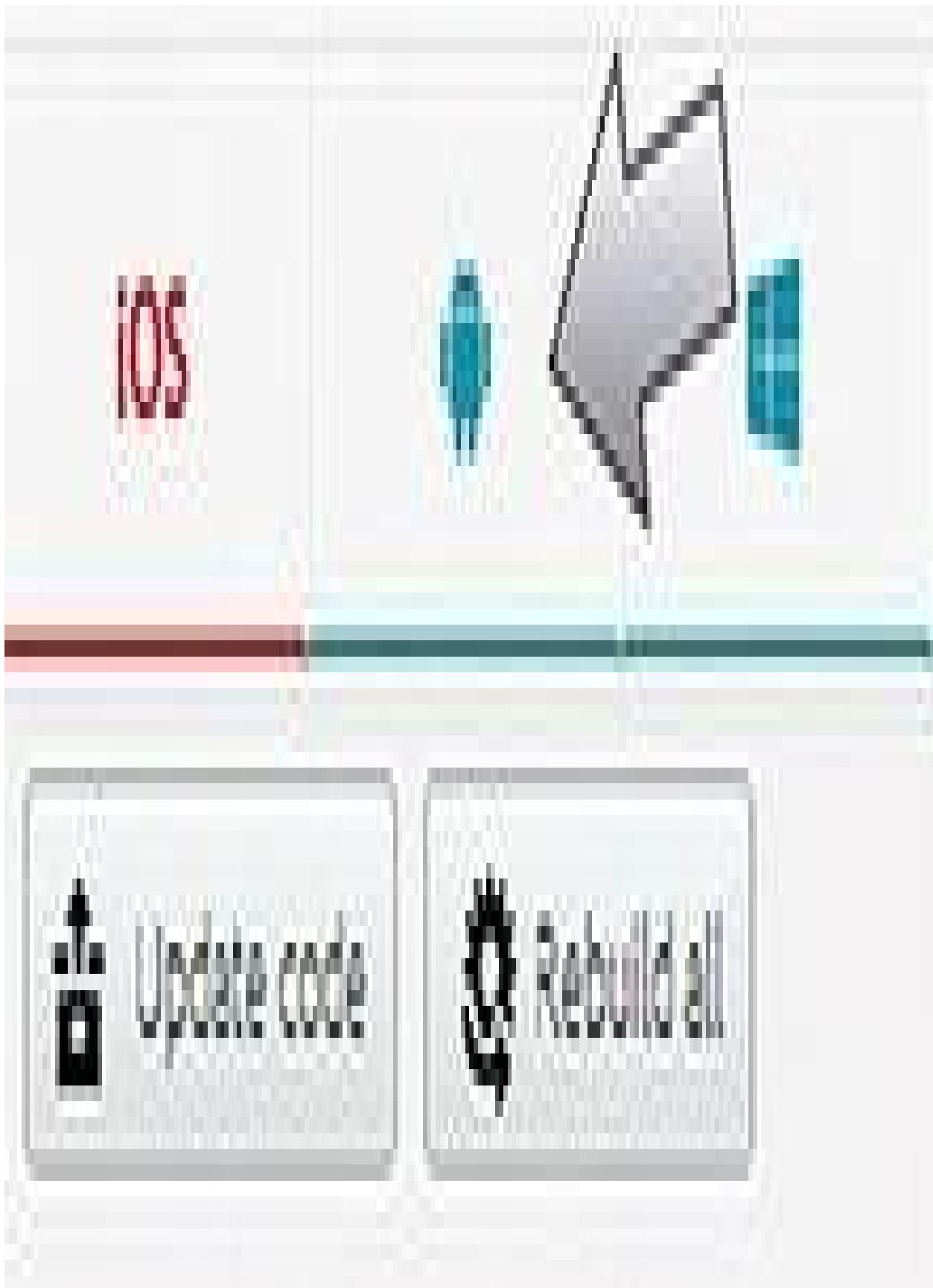
### **Figura 246**

Para baixar o aplicativo diretamente para o seu dispositivo móvel, utilize o QR Code gerado automaticamente na página (Figura 247).



### **Figura 247**

Se preferir, clique no botão “Android” (Figura 248) para baixar o arquivo de instalação. Para instalar o aplicativo em seu dispositivo Android, copie o arquivo para o dispositivo e o execute.



### **Figura 248**

Veja na Figura 249 o aplicativo instalado em um dispositivo Android.



**Figura 249**

## Conclusão

Finalizamos o conteúdo deste livro. É claro que o HTML5 oferece muito mais recursos para o desenvolvimento de jogos, mas espero que este livro tenha despertado em você a curiosidade de buscar mais informações e conhecimentos para o desenvolvimento de seus jogos. Caso você tenha alguma dúvida ou sugestão, poderá entrar em contato comigo pelo e-mail  [contato@denilsonbonatti.com.br](mailto: contato@denilsonbonatti.com.br) ou diretamente pelo site [www.denilsonbonatti.com.br](http://www.denilsonbonatti.com.br).

Obrigado.