UNIVERSIDADE FEDERAL DO AMAZONAS
PRO REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO À PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

O CUSTO COMPUTACIONAL DAS ABORDAGENS AD-HOC PARA A CONSTRUÇÃO DE BORDAS ARREDONDADAS UTILIZANDO TECNOLOGIA WEB EMBARCADA EM PLATAFORMAS MÓVEIS

Fábio Martins Lima
Bolsista - CNPQ

MANAUS
2011

UNIVERSIDADE FEDERAL DO AMAZONAS
PRO REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO À PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

RELATÓRIO FINAL
PIB-E/0022/2010

O CUSTO COMPUTACIONAL DAS ABORDAGENS AD-HOC PARA A CONSTRUÇÃO DE BORDAS ARREDONDADAS UTILIZANDO TECNOLOGIA WEB EMBARCADA EM PLATAFORMAS MÓVEIS

BOLSISTA: Fábio Martins Lima
ORIENTADOR: Prof. Dr. César Augusto Viana Melo

MANAUS
2011

Esta pesquisa, financiada pelo Conselho Nacional de Pesquisa – CNPQ, através do Programa Institucional de Bolsas de Iniciação Científica da Universidade Federal do Amazonas, foi desenvolvida no Departamento de Ciência da Computação.

RESUMO

No mercado das aplicações móveis, construir uma aplicação com funcionalidades inovadoras e eficientes não é garantia de sucesso. Neste mercado, a construção de aplicativos com grande apelo visual tem ganhado cada vez mais importância. O conceito de bordas arredondadas é um elemento chave no processo de construção dessas interfaces, com diversos métodos tendo sido relatado na literatura. Esses métodos apresentam funcionalidades semelhantes, porém possuem demandas de processamento distintas. Neste projeto analisam-se os custos computacionais dos diversos métodos de construção de bordas arredondadas existentes, para que se tenha conhecimento dos aspectos favoráveis de cada um.

Palavras-chave: WRT, *programação WEB, Webkit, renderização, análise assintótica*.

SUMÁRIO

1. Introdução

O conceito de bordas arredondadas é um elemento chave no processo de construção de interfaces, afinal as aplicações atualmente possuem um grande apelo visual. Contudo, as plataformas de desenvolvimento baseadas em tecnologias WEB, em especial a plataforma de desenvolvimento WRT considerada neste trabalho, não dispõem de ferramental nativo para tratamento deste conceito. Assim, esta lacuna tem desafiado os projetistas e programadores a desenvolverem abordagens alternativas com uma série de métodos de programação.

Existem métodos de arredondamento de bordas com funcionalidades parecidas, mas que possuem características de processamento distintas.

Neste projeto, analisam-se os custos computacionais dos métodos de arredondamento de bordas existentes, para que, cada método possa ser usado na ocasião em que apresenta melhor desempenho.

O restante deste está organizado da seguinte maneira: Na Seção 2, apresenta-se a revisão bibliográfica realizada até o momento, tendo-se estudado as Tecnologias WEB, a Engine Webkit e a análise assintótica de algoritmos. Métodos utilizados são apresentados na Seção 3, os resultados dos são apresentados na Seção 4, seguido pela conclusão na Seção 5.

2. Revisão Bibliográfica

Como base para o estudo dos métodos para arredondamento de bordas nas aplicações para aparelho móvel, os seguintes temas mostraram-se importantes: Tecnologias WEB, Engine WebKit e Análise Assintótica de Algoritmos.

2.1 Tecnologias WEB

Para implementação na plataforma WebRuntime, utilizam-se as tecnologias de programação para Internet que são: HTML, CSS e Javascript.

Explorando essas tecnologias em conjunto, adquire-se ferramenta para o desenvolvimento de diversas funcionalidades para o programa e também para a interface.

Cada tecnologia apresenta características específicas para a criação de programas para plataformas móveis (Figura 2.1).

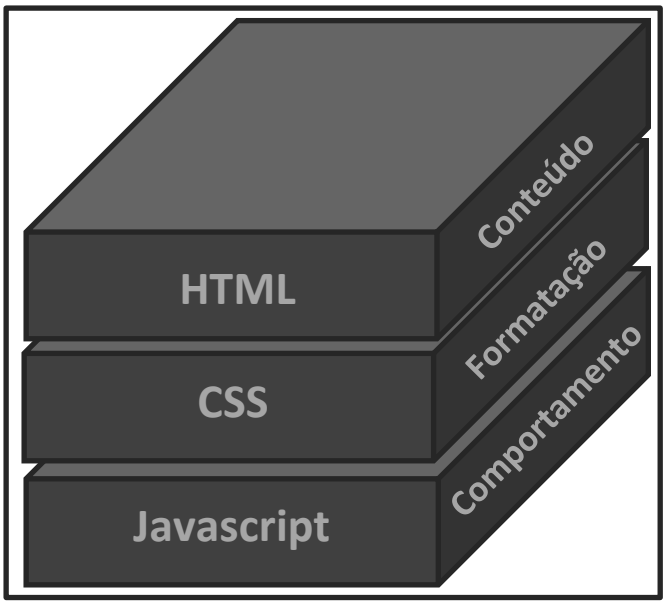

Figura 2.1 – As tecnologias WEB e suas principais características.

O HTML é responsável pela criação do que será impresso na tela do aparelho, através de “marcadores”, que são palavras-chave que indicam a forma renderizada na tela.

O CSS é responsável pelo estilo que o “marcador” do HTML deve tomar, modificando as características do marcador, ou gerando uma classe a qual esse marcador pertencerá, esta última opção proporciona ao programador a vantagem de não precisar redefinir estilos semelhantes aos que foram usados em outro momento na aplicação.

O uso primário do Javascript é escrever funções que são embarcadas ou incluídas na codificação HTML e que interagem com o Document Object Model(DOM), funcionando diretamente no navegador do usuário, deste modo, o navegador pode responder às ações realizadas pelo usuário rapidamente, criando uma aplicação mais iterativa.

2.2 Engine Webkit

A renderização de uma página, na plataforma WRT é realizada pelo Engine Webkit que é subdividido em três componentes: WebCore, JavaScriptCore e Drosera.

O WebCore gera o DOM (sigla em inglês para Modelo de Objeto do Documento) para o HTML, ou seja, ele é responsável pela renderização de imagens embutidas no documento, entre outras coisas. O JavaScriptCore é uma estrutura que fornece um mecanismo JavaScript para implementações Webkit. O Drosera é um depurador de JavaScript presente no Webkit que facilita a escrita do código, pois ajuda a evitar erros.

A carga de uma página HTML é realizada por dois módulos que estão presentes na Engine Webkit, são eles: FrameLoader e DocLoader.

FrameLoader. Encarrega-se de carregar os documentos. Sempre que um link é clicado o FrameLoader começa criando um novo objeto (que chamamos de DocumentLoader), onde aguarda uma decisão do cliente Webkit, sobre como ele deverá lidar com este objeto.

DocLoader. Uma página WEB requer mais do que apenas o HTML que inclui o documento, precisa carregar também imagens, scripts e outros sub-recursos referenciados pelo documento. Função do DocLoader.

2.3 Análise assintótica de algoritmos

Os diferentes algoritmos geradores de bordas arredondadas, do ponto de vista do usuário, produzem resultados muito próximos, sendo as diferenças imperceptíveis aos olhos humanos. Para diferenciar estes algoritmos utiliza-se uma das três principais classificações: Ordem O, Ordem Omega e Ordem Theta.

Na análise realizada com a Ordem O (Figura 2.2(b)), estabelece-se um limite superior para uma família de funções, sendo esse limite definido da seguinte maneira:

$O(g(n)) = \{f(n)$: existem constantes positivas c e n_0 tais que

$0 \leq f(n) \leq cg(n)$ para todo $n \geq n_0$.

Na análise realizada com a Ordem Ω(Figura 2.2(c)), estabelece-se um limite inferior da seguinte maneira:

$\Omega\ (g(n)) = \{f(n)$: existem constantes positivas c e n_0 tais que

$0 \leq cg(n) \leq f(n)$ para todo $n \geq n_0$.

A Ordem θ(Figura 2.2(a)), une as Ordens O e Ω, estabelecendo assim, um limite inferior e superior, assim:

θ *(g(n))* = {*ƒ(n)* : existem constantes positivas c_1, c_2 e n_0 tais que

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0$.

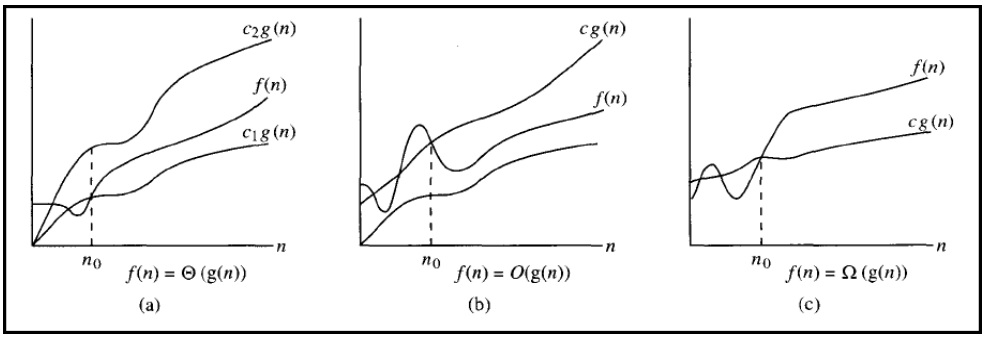

Figura 2.2 – Exemplos gráficos das notações θ, O e Ω.

3. Métodos Utilizados

3.1 Análise Assintótica e o Custo dos Algoritmos

Inicialmente, mostrou-se necessário um estudo a respeito da maneira em que cada método de arredondamento de bordas funciona na aplicação. Deste modo, antes de iniciar testes com o aparelho, estabeleceu-se o custo correspondente a cada um dos métodos utilizando a análise assintótica de algoritmos.

Cada método possui uma estrutura diferente, sendo assim, os custos dos métodos podem ser diferentes, contudo, assintoticamente esses custos podem ser semelhantes. Sendo assim, os métodos podem apresentar desempenho semelhante ou não nos diferentes cenários em que podem ser aplicados.

3.2 Testes com aplicações

Para verificar a precisão dos cálculos teóricos, realizaram-se testes por meio de aplicações do aparelho móvel.

Os testes consistem em realizar transições de imagens no visor do aparelho, utilizando o mesmo método de arredondamento de bordas. Assim, através de um critério de parada, pode-se através do número de iterações, obter dados relevantes para construção de tabelas e gráficos comparativos em relação ao cálculo teórico.

4. Resultados e Discussão

4.1 O custo dos métodos estudados

Em programação utiliza-se o número de comparações realizadas para definir o custo de um algoritmo. Contudo, nem todos os métodos de arredondamento de bordas utilizam codificação, sendo assim, esse critério não poderia ser utilizado no cálculo de custo de todos os métodos, pois se necessita de um critério presente em ambos para que se realizem comparações.

Como objeto de estudo na análise utilizou-se o número de pixels renderizados de cada método já que este objeto é totalmente relevante ao estudo além de poder ser verificado experimentalmente.

4.1.1 Método utilizando codificação

Este método utiliza formas criadas pelo HTML e estilizadas pelo CSS, criando camadas de um pixel nas partes superior e inferior da figura a ser renderizada, gerando assim as bordas.

O número de camadas que uma figura terá quando aplicado o método é de 8 camadas de pixels a mais (4 acima e 4 abaixo do bloco central da imagem). Portanto um caso mínimo para o método seria de uma imagem com 9 pixels de altura e 9 de largura (Figura 4.1).

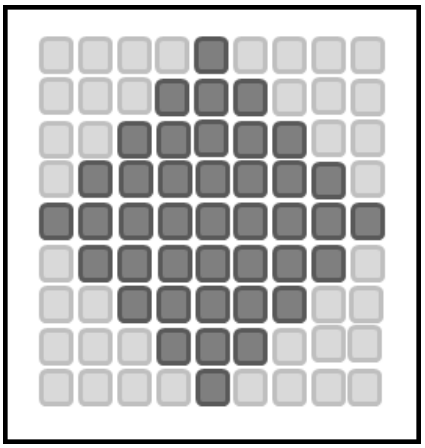

Figura 4.1 – Pixels renderizados (blocos escuros) em método utilizando codificação.

Os pixels que não são renderizados devem ser adicionados ao cálculo de custo. Utilizando o método de codificação, 40 (quarenta) pixels nunca serão renderizados. Este fato depende da escrita da classe que manipula a criação das bordas nas figuras, podendo aumentar caso se deseje aperfeiçoar a curvatura da borda. Contudo, a definição utilizada nos testes é a apresentada na Figura 4.1.

O custo em termos de pixels é dado pela altura e largura da imagem, que representam o número total de pixel desta. Porém realiza-se o desconto referente aos pixels que o método não renderiza para modelar a borda.

$$f(n) = altura \times largura - 40$$

Representando altura x largura como o número de pixels (n):

$$f(n) = n - 40$$

Assintoticamente, o custo para o método seria θ(n), pois os valores constantes em uma análise são desprezados, já que quanto mais o valor de n cresce torna-se desprezível a influência da constante do custo final do método.

Porém, é necessário que se realize uma prova para a verificação de tal afirmação. Para tal utilizou-se a definição de θ:

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0,$$

$$\text{tais que } 0 \leq c_1 g(h) \leq f(n) \leq c_2 g(n) \ \forall \ n > n_0\}$$

Sendo assim, a inequação de custo se torna:

$$\mathbf{0 \leq c_1 n \leq n - 40 \leq c_2 n}$$

Devemos isolar as constantes, portanto divide-se toda a inequação por n:

$$\mathbf{0 \leq c_1 \leq 1 - 40/n \leq c_2}$$

Agora deve-se escolher um valor pra n_0 que seja sempre menor que n, sabemos que n mínimo é 81 (quanto temos 9 linhas e 9 colunas), então podemos utilizar $n_0 = 80$:

$$\mathbf{0 \leq c_1 \leq 0{,}5 \leq c_2}$$

Escolhendo $c_1 = 0{,}2$ e $c_2 = 1$, temos:

$$\mathbf{0 \leq 0{,}2 \leq 0{,}5 \leq 1}$$

Portanto para todo $n \geq 80$, temos que:

$$\mathbf{0 \leq 0{,}2n \leq n - 40 \leq n}$$

Assim, conclui-se que o custo do método relacionado ao número de pixels renderizados é de **θ(n)**.

4.1.2 Método utilizando imagens

Ao criar a imagem em um editor pode-se especificar o tamanho dela em termos de altura e largura, assim a imagem preenche todo o espaço reservado

para ela. Entretanto, diferente do método com codificação, quando se exclui a parte da borda, esse trecho invisível continua a fazer parte da imagem, portanto:

$$f(n) = altura*largura \rightarrow f(n) = n$$

Contudo, é relevante que, segundo o estudo da Engine Webkit, para uma imagem ser renderizada, ela deve passar por pelo módulo DocLoader, que é o responsável no navegador para carregar os subrecursos da aplicação (no caso as imagens). Com isso existe um custo adicional para que seja visualizado um pixel na tela do computador.

Não se pode determinar o valor específico para a renderização de um pixel de uma imagem, pois isso varia de acordo com a velocidade de processamento. Contudo, sabe-se que carregar uma imagem presente nos arquivos da aplicação apresenta um custo adicional.

Portando a função de custo é dada por:

*f(n) = n*x, sendo x o custo adicional para carregar uma imagem.*

O custo para gerar uma imagem é maior que o de apenas usar codificação para geração de apenas uma imagem também, então x>1.

Para uma constante multiplicada pela instância temos que o custo, assim como no método de codificação seria θ(n).

Para provar podemos utilizar novamente definição de θ(n):

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0,$$

$$\text{tais que } 0 \leq c_1 g(h) \leq f(n) \leq c_2 g(n) \ \forall \ n > n_0\}$$

Sendo assim, a inequação de custo se torna:

$$\mathbf{0 \leq c_1 n \leq n*x \leq c_2 n}$$

Devemos isolar as constantes, divide-se então toda a inequação por n:

$$\mathbf{0 \leq c_1 \leq x \leq c_2}$$

O custo passa a ser representado por x para qualquer n_0 que seja escolhido, com isso, podemos escolher valores de c_2 que podem satisfazer a inequação;

Escolhendo c_1 =a e c_2 = x+1, tal que 0<a<x.

$$0 \leq a \leq x \leq x+1$$

Portanto para todo n, temos que:

$$0 \leq a*n \leq n*x \leq (x+1)*n$$

Assim, conclui-se que o custo do método utilizando imagens também é de **θ(n)**.

4.1.3 Método utilizando imagens e codificação

Este método, assim como o que utiliza imagens, requer o carregamento de imagem. Contudo essa imagem possui uma menor dimensão quando relacionado ao outro método, afinal esta é utilizada apenas para modelagem da borda. O corpo da figura é estilizado pelo CSS que cria o aspecto de que uma única imagem está sendo utilizada. A distribuição de pixels pode ser

visualizada na Figura 4.2, onde os pixels mais escuros representam a codificação, os pixels restantes representam as figuras.

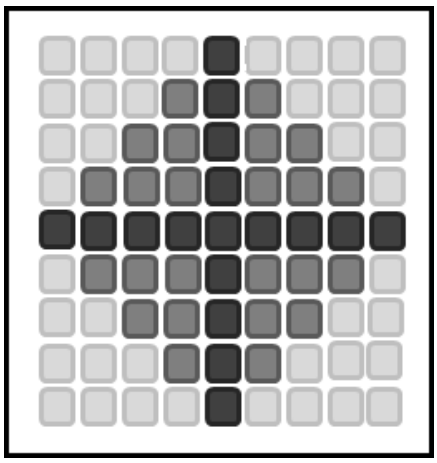

Figura 4.2 – Representação em pixels de uma figura com método de figuras e codificação.

Para cada borda, tem-se 16 pixels renderizados com imagem.

Assim, 64 pixels possuem custo de carregamento.

*f(n) = altura*largura = n*

Descontando os pixels com custo a mais:

f(n) = n + 64x

Para uma constante multiplicada pela instância temos que o custo seria θ(n).

Para provar podemos utilizar novamente definição de θ(n):

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \leq c_1 g(h) \leq f(n) \leq c_2 g(n)\ \forall\ n > n_0\}$$

Sendo assim, a inequação de custo se torna:

$0 \leq c_1 n \leq n + 64x \leq c_2 n$

Devemos isolar as constantes, portanto divide-se a inequação por n:

$$\mathbf{0 \leq c_1 \leq 1 + 64x/n \leq c_2}$$

Deve-se escolher um valor pra n_0 que seja sempre menor que n, sabemos que n mínimo é 81 (quanto temos 9 linhas e 9 colunas), então podemos utilizar $n_0 = 80$:

$$\mathbf{0 \leq c_1 \leq 1 + 64x/80 \leq c_2}$$

$$\mathbf{0 \leq c_1 \leq 1 + 0{,}8x \leq c_2}$$

Considerando um valor de x=2, já que ele é maior que 1

$$\mathbf{0 \leq c_1 \leq 1 + 1{,}6 \leq c_2}$$

$$\mathbf{0 \leq c_1 \leq 2{,}6 \leq c_2}$$

Escolhendo $c_1 = 2$ $c_2 = 3$

$$\mathbf{0 \leq 2 \leq 2{,}6 \leq 3}$$

Portanto para todo n>80 com x sendo maior que 1, temos que:

$$\mathbf{0 \leq 2n \leq n + 64x \leq 3n}$$

Assim, conclui-se que o custo do método utilizando imagens também é de $\boldsymbol{\theta(n)}$.

O custo dos três métodos mostrou-se semelhante quando a análise assintótica, afinal o critério de comparação era o número de pixels, quantidade essa que possui valor idêntico para os métodos analisados. Contudo, algumas diferenças já puderam ser notadas como a soma de constantes para renderização de imagens. Outro fator importante para distinção entre os métodos é o custo que a Engine Webkit atribui a renderização de cada marcador HTML, esse fator é importante, pois quanto maior o número de

marcadores mais custo é adicionado ao custo final. Assim como o custo do carregamento de imagens, a renderização dos marcadores depende da capacidade de processamento que o aparelho, para isso tornam-se necessários testes experimentais para verificação destes valores.

4.2 Aplicações testes

Para validação da análise assintótica, assim como obtenção de dados a respeito da diferença dos valores constantes relacionados ao carregamento de imagens e marcadores HTML realizaram-se testes que consistiam na transição de botões no visor do aparelho, utilizando a cada teste um método de arredondamento de bordas.

Incialmente, o processo foi realizado com o critério de parada sendo o nível de bateria do aparelho. Contudo o teste não se realizou com sucesso no aparelho, por a queda de carga não ocorria por unidade e sim por níveis (por exemplo, de 25 em 25%). De qualquer maneiro ocorreram tentativas de realizar o teste nesse cenário, contudo o aparelho apresenta limitações quanto ao uso de memória, impedindo que a aplicação continuasse a ser executada até que se chegasse ao momento em que ocorresse a queda no nível da bateria. Da mesma forma, o processo não pôde ser realizado no simulador pois a simulação da carga da bateria era realizada manualmente.

Contudo, se fossem limitados o número de iterações que a aplicação iria realizar, seria possível obter o número de pixels renderizados em relação ao tempo em que realizara as transições. Assim como o teste anterior, as limitações de memória impediram que o teste fosse realizado no aparelho para

grandes instâncias e, como não foi possível obter informações relevantes a partir de dados pequenos, o teste foi realizado via simulador.

Através dos testes realizados obteve-se o seguinte gráfico representativo:

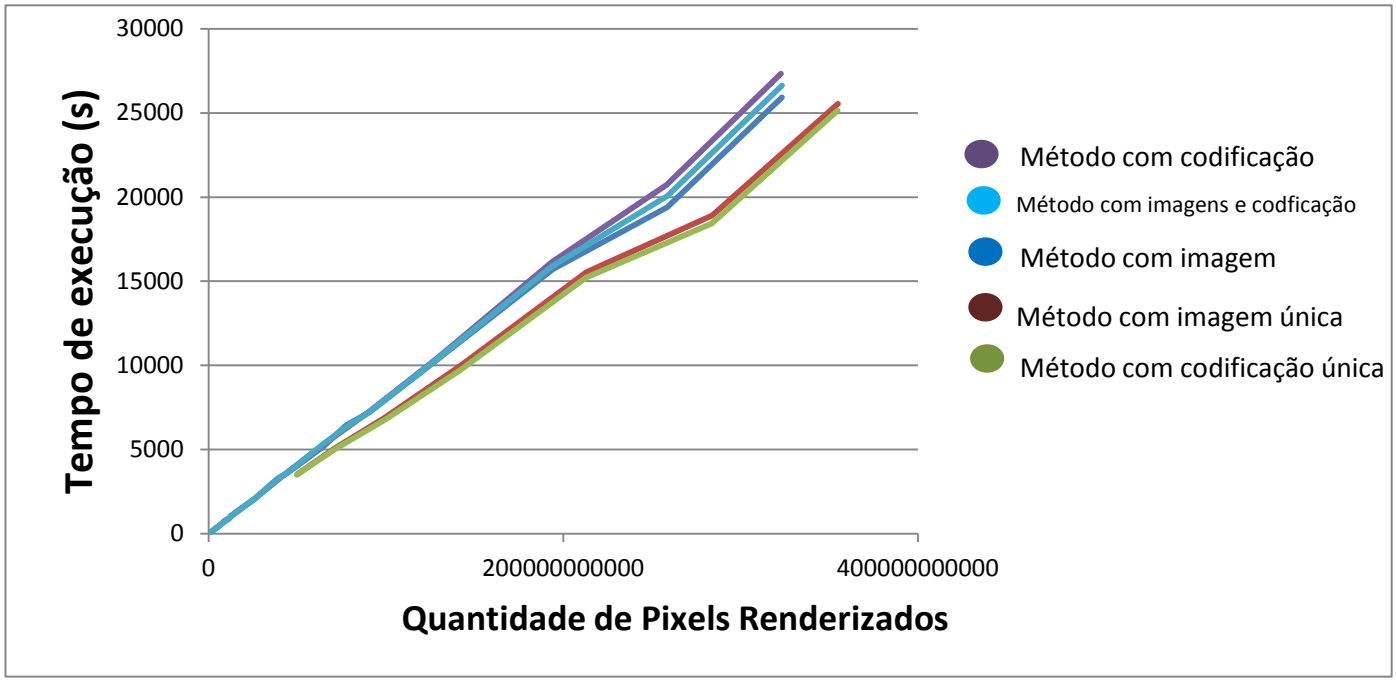

Figura 4.3 – Representação em pixels de figuras com método de imagem e codificação.

Segundo os dados, os custos dos métodos são realmente semelhantes e, que a quantidade de imagens que serão renderizadas por view da aplicação interfere diretamente em relação ao custo.

5. Conclusão

Os métodos estudados apresentam custo semelhante, portanto, o que os diferencia é apenas o cenário em que cada um será utilizado.

Em uma aplicação de pequeno porte, todos os métodos apresentam um desempenho semelhante, contudo alguns aspectos podem ser relevantes na escolha. Por exemplo, em uma aplicação que requer um forte apelo visual sem limitações quanto ao espaço ocupado pelo aplicativo no aparelho é recomendado o uso de imagens, afinal, com o auxílio de uma ferramenta de edição de imagens, pode criar com mais facilidade um visual mais sofisticado. Em contrapartida, quando se deseja utilizar uma aplicação que ocupe um espaço menor recomenda-se o uso de codificação.

Quando se quer um menu onde todos os botões são iguais, recomenda-se o uso de imagens, pois, após a imagem ser carregada, ele será armazena em cash, facilitando assim a segunda chamada da mesma, enquanto que o custo de renderizar os marcadores HTML deveriam ser realizados para cada botão do menu.

Em uma aplicação de grande porte, não é recomendado o uso excessivo de imagens, pois para da imagem, é necessário uma espaço em disco para armazená-la o que pode deixar a aplicação com um tamanho maior, o que não é uma característica boa para aplicações em aparelhos móveis, que não possuem alta capacidade de armazenamento. Sugere-se que imagens sejam utilizadas em certos pontos ou ainda em casos que a mesma repita-se em várias instâncias.

6. Referências Bibliográficas

[1] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C.**Algoritmos Teoria e Prática**. Tradução da 2ª edição americana. São Paulo: Campus, 2002. 916. São Paulo.. Elsevier, 2002.

[2] FREEMAN, E.; FREEMAN, E. **Use a cabeça – Html com CSS e XHtml**. 1ª edição. São Paulo, 2008. Altabooks.

[3] MCFARLAND; SAWYER, D. **Css – O manual que faltava**. 1ª edição. São Paulo, 2007. Digerati.

[4] SILVA; SAMY, M. **Javascript, guia do programador**, 1ª edição. São Paulo, 2008. Novatec.

[5] DAMIANI; EDGARD, B. **Javascript – Guia de consulta Rápida**, 3 ª edição. São Paulo, 2008. Novatec.

[6] **Fórum Nokia.** Disponível em: http://www.forum.nokia.com. Acesso em: Agosto de 2010.

[7] **W3Shcool.** Disponível em: http://www.w3school.org. Acesso em: Setembro de 2010.

[8] **WebKit.org.** Disponível em: http://webkit.org. Acesso em: Outubro de 2010.

[9] **W3C**. Disponível em: http://maujor.com/w3ctuto/roundshadow.html. Acesso em: Novembro de 2010.

[10] **Web Neurotic**. Disponível em: http://www.neuroticweb.com/css-box/. Acesso em: Novembro de 2010

[11] **WorldLingo**. Disponível em: http://www.worldlingo.com. Acesso em: Dezembro de 2010.

7. Cronograma de Atividades

Nº	Descrição	Ago 2010	Set	Out	Nov	Dez	Jan 2011	Fev	Mar	Abr	Mai	Jun	Jul
1	Estudo das tecnologias web para desenvolvimento de aplicações móveis.	R	R	R									
2	Estudo do engine Webkit			R	R	R							
3	Estudo da análise assintótica de algoritmos				R	R	R						
4	Estudo de métricas de avaliação de custo computacional em plataformas móveis						R	R	R				
5	Implementações de Estudos de caso envolvendo as técnicas de construção de interfaces gráficas baseada em tecnologias Web									R	R		
6	Avaliação da aplicação implementada para estudo de caso										R		
7	Elaboração do Resumo e Relatório Final											R	R
8	Preparação da Apresentação Final para o Congresso												R
9	Reuniões periódicas com o orientador	R	R	R	R	R	R	R	R	R	R	R	R