Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Ronaldo Yang
Yoshio Mori

Desenvolvimento de um baralho para dispositivos móveis Android

São Paulo
Novembro de 2016

Desenvolvimento de um baralho para dispositivos móveis Android

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Flávio Soares Corrêa da Silva

São Paulo
Novembro de 2016

Resumo

Desde os primeiros dispositivos móveis com interface gráfica até os atuais *Smartphones* e *Tablets*, os *games mobile* tiveram um grande papel na popularização desses aparelhos. Com isso, surgiram diversos gêneros distintos de jogos, dentre os quais, aqueles que simulam passatempos reais.

Como jogos de cartas são um dos entretenimentos mais difundidos na sociedade moderna, pretendemos, neste trabalho, desenvolver um aplicativo para *Android* de um simulador de baralho, de modo a ser utilizado em qualquer partida local como um substituto de um baralho real. Para tal, existirá um dispositivo à parte, representando a "mesa", onde os jogadores devem se conectar, formando um modelo cliente-servidor. Após as configurações de redes estiverem sido estabelecidas, os participantes poderão realizar ações básicas de um jogo de baralho, como embaralhar, mover e enviar cartas.

Nessa aplicação utilizaremos apenas bibliotecas disponíveis pela *API* do *Android* através de seu *SDK*. Entre elas, usaremos *OpenGL ES* para implementação gráfica e *Wi-fi Direct* para redes.

Palavras-chave: Jogos de Cartas, Baralho, Simuladores, Jogos *Mobile*, *Android*, *OpenGL ES*, *Wi-fi Direct*.

Abstract

From the first mobile devices with graphical user interface to the current Smartphones and Tablets, mobile games had a great role in popularizing these devices. Therefore, several different genres of games appeared, among which, those that simulate real pastimes.

As playing cards are one of the most widespread entertainments in modern society, we intend, in this work, to develop a deck of cards simulator application for Android, so that it can be used in any local game as a substitute for a real deck of cards. In order to achieve this, there will be a separate device, representing the " table ", where players must connect, forming a client-server model. After the network settings have been established, participants can perform basic actions of a playing card game, such as shuffling, moving and sending cards.

In this application we will only use libraries available through Android SDK. Among them, we will use OpenGL ES for graphical and Wi-Fi Direct for networks implementations.

Keywords: Playing Card, Deck of Cards, Simulators, Mobile Games, Android, OpenGL ES, Wi-fi Direct.

Sumário

Capítulo 1

Introdução

O desenvolvimento de um aplicativo android nos remete aos computadores dos anos 90 com modem de internet, cpu e memória limitados e tudo isso sendo alimentado por uma pequena bateria. Neste projeto iremos desenvolver um aplicativo android de um baralho virtual para ser usado em rede local como um substitudo de um baralho real. Com um baralho real, podemos misturar, virar, desvirar e distribuir as cartas e serão essas as ações que nosso aplicativo deve fazer. Para misturar as cartas o usuário deve selecionar todas as cartas mantendo o dedo sobre as cartas e arrastar as cartas para cima e para baixo algumas vezes. Para virar e desvirar o usuário deve dar um duplo toque sobre as cartas, se várias cartas foram selecionadas, todas as cartas selecionadas serão viradas ou desviradas. Para distribuir as cartas, o usuário deve arrastar a carta até a borda da tela em direção do jogador desejado. Iremos ver que o toque sobre a imagem de um objeto e a interpretação do movimento do dedo na tela são problemas no desenvolvimento deste tipo de aplicativos.

O aplicativo será desenvolvido para aparelhos com sistema operacional android versão 5.0.1 com o sistema gráfico OpenGL ES. Android é um sistema operacional desenvolvido pela Google para dispositivos com telas sensíveis ao toque tais como smartphone e tablet. Como os smartphone e tablet possuem telas que variam muito de tamanho, um dos problemas no desenvolvimento deste aplicativo foi adaptar as imagens dos objetos para a tela. O sistema gráfico OpenGL ES é uma interface de software para hardware gráficos. A interface consiste de um conjunto de procedimentos e funções que nos permitem especificar objetos e operações para produzir imagens gráficas de alta qualidade.

Em adicional, queremos especificar uma linguagem para possibilitar o aplicativo executar essas ações de forma automática e também queremos implementar a detecção automática da direção entre os dispositivos para que as cartas sejam passadas de um dispositivo para outro com um toque e arrastar dos dedos.

Capítulo 2

Desenhando objetos

Para desenhar imagens de cartas, fundo e botões na tela do celular, fizemos uso da interface de programação de aplicações (API) do Android e do OpenGL.

O Android pode ser visto como uma camada de abstração da interface de manipulação da tela do celular e do sensor de toque da tela disponibilisados pelos fabricantes. Nosso aplicativo faz uso desses dispositivos, por meio desta camada e não iremos nos preocupar como essa abstração foi implementada.

Um aplicativo para Android é construido com componetes. Cada componete é um diferente ponto de entrada do sistema. Existem quatro tipos diferentes de componetes: Activities, Services, Content providers e Broadcast receivers, cada qual responsável por alguma funcionalidade dentro do aplicativo.

Nesta etapa de desenvolvimento, faremos uso da componete Activity. Uma activity representa uma tela única com uma interface de usuário. Afim de gerar imagens para serem desenhadas nesta tela, faremos uso do OpenGL ES.

O OpenGL é um conjunto de comandos que permitem a especificação de objetos geométricos em duas ou três dimenções, junto com comandos que controlam como esses objetos são renderizados no *framebuffer*. Esses objetos geométricos são construidos com *primitivas* que podem ser ponto, segmento de linha, ou triângulo. As primitivas são definidas por um grupo de um ou mais vértices. Os vértices definem um ponto, ponto final de uma aresta ou canto de um triângulo onde duas arestas se encontram e são representados como um subconjunto de $\mathbb{R}^2$. Informações como coordenadas de posição, cores, normal, coordenadas de textura podem estar associados a um vértice. As primitivas formadas pelos vértices em $\{(x,y) \in \mathbb{R}^2 | -1 \leq x,y \leq 1\}$ são enviados para serem mostrados na tela, chamaremos esses vértices de visíveis.

Inicialmente haviamos escolhido um modelo 3d com 312 vértices para desenhar a imagem de carta. No teste de desempenho usando o celular da lg modelo F240L, o aplicativo estava gerando 8 frames por segundo. Decidimos usar um modelo 2d com 6 vértices, como resultado o aplicativo passou a gerar 71 frames por segundo.

Existem muitos tipos e modelos de aparelhos mobile e as dimenções da tela podem variar muito de modelo para modelo, o que causa uma distorção na imagem desenhada na tela. Queremos que nosso aplicativo seja capaz de fazer essa correção automaticamente, de tal modo que todos os vértices visíveis continuem sendo visíveis após a correção.

Sejam respectivamente w, h o comprimento e altura de uma tela, a transformação viewport pode ser descrita por uma transformação $V_{w,h}$ que leva elementos de $\mathbb{R}^2$ para $\mathbb{R}^2$ tal que $(x,y) \mapsto \left(\frac{wx}{2}, \frac{hy}{2}\right)$.

Dizemos que D transformação que leva elementos de $\mathbb{R}^2$ para $\mathbb{R}^2$ é distorção se existem $(x,y), (x',y') \in \mathbb{R}^2$ tal que $x = y$, $D(x,y) = (x',y')$ e $x' \neq y'$. Observe que se $w \neq h$ então

$V_{w,h}$ é distorção.

Dizemos que C transformação que leva elementos de $\mathbb{R}^2$ para $\mathbb{R}^2$ é correção da distorção D se para todo $(x,y),(x',y') \in \mathbb{R}^2$ tal que $x = y$ e $D \circ T(x,y) = (x',y')$, temos $x' = y'$.

Iremos mostrar que

$$T(x,y) = \begin{cases} \left(\frac{hx}{w}, y\right) & \text{se } w > h \\ \left(x, \frac{wy}{h}\right) & \text{c.c.} \end{cases}$$

é correção da $V_{w,h}$ e que se o vértice (x,y) é visível então $T(x,y)$ é visível.

Seja $a \in \mathbb{R}$ e $(x,y) \in \mathbb{R}^2$ tal que $x = y = a$. Se $w > h$, então $V_{w,h} \circ T(x,y) = \left(\frac{ha}{2}, \frac{ha}{2}\right)$. Se $w \leq h$, então $V_{w,h} \circ T(x,y) = \left(\frac{wa}{2}, \frac{wa}{2}\right)$.

Seja $(x,y) \in \mathbb{R}^2$ visível. Se $w > h$ então $T(x,y) = \left(\frac{hx}{w}, y\right)$ e temos $-1 \leq \left(\frac{hx}{w}, y\right) \leq 1$. Se $w \leq h$ então $T(x,y) = \left(x, \frac{wy}{h}\right)$ e temos $-1 \leq \left(x, \frac{wy}{h}\right) \leq 1$.

Aplicamos a transformação T nos vértices no vertex shader e obtemos uma imagem sem deformação e com todos os vértices que deviam ser mostrados na tela.

Para mover as cartas, o usuário deve tocar na carta e arrastá-la. A API do Android nos permite saber a posição do toque na tela, mas ainda precisamos saber se existe alguma carta nesta posição. Existem duas soluções para este problema.

A primeira solução consiste em gerar uma imagem em preto e branco para cada carta, onde a carta é desenhada em branco no fundo preto. Para verificar que a posição do toque contem uma carta, basta ler o pixel correspondente à posição do toque. Se o pixel for branco, então a carta foi tocada. Contudo, o acesso ao framebuffer para leitura do valor do pixel é restrito apenas para o sistema e não pudemos implementar esta solução.

Assim, passamos para a segunda solução. Aplicaremos a transformação inversar da $V_{w,h} \circ T(x,y)$ nas coordenadas do toque para obtermos um ponto na mesma coordenada dos vértices que geram a imagem da carta. Dessa forma podemos verificar se o ponto está entre os vértices da carta.

Para embaralhar, o usuário deve selecionar todas as cartas arrastar para cima e para baixo pelo menos 3 vezes. Assim, temos duas classes de movimento, uma para embaralhar e outra só para movimentar. A solução para este problema foi aplicar o classificador de k vizinhos próximos que é um método para resolver problemas de aprendizagem supervisionada.

O classificador de k vizinhos mais próximos recebe X conjunto de entrada e Y conjunto de saída tais que a i-ésima observação de $x_i \in X$ está associado ao i-ésimo elemento $y_i \in Y$. chamaremos o conjunto de todas as associações entre X e Y de conjunto de treinamento.

Seja x O método é fazer observações nos k elementos mais próximo à x nova entrada do conjunto de entrada.

Capítulo 3

Conclusões

Texto texto[1].

[1]Exemplo de referência para página Web: www.vision.ime.usp.br/~jmena/stuff/tese-exemplo

Apêndice A

Título do apêndice

Texto texto.