UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO PRÁTICO FINAL

HÉLIO CARLOS BRAUNER FILHO - 180182
JOÃO LUIZ GRAVE GROSS - 180171

Trabalho da Disciplina Modelos de Linguagens de Programação

Prof. Leandro Krug Wives

Porto Alegre, 21 de junho de 2012.

1 Apresentação

1.1 Características do programa

A proposta do trabalho é a criação de um aplicativo para celular Android, um jogo em que o usuário controla um morcego que deve comer frutas que são geradas na tela e se movimentam. O objetivo é avançar o maior número de fases possível dentro de um tempo determinado, no caso, dois minutos. A cada fase, o número de frutas que precisa ser comido aumenta, bem como a velocidade com que as frutas se movimentam. O morcego é controlado pelo toque do dedo sobre ele, e as frutas são comidas quando o gráfico do morcego se encontra com o gráfico das frutas.

1.2 API e bibliotecas

Primeiramente, definimos que o jogo seria criado para a plataforma Android, e, para desenvolver, seria usada a linguagem Java 7 com a API do SDK Android. O SDK Android possui diversas classes diferentes, voltadas principalmente para a geração de objetos na tela do celular e a integração do aplicativo com o sistema. Estas classes podem ser facilmente estendidas ou personalizadas.

Depois, tendo em vista as especificações deste trabalho, pensamos em utilizar uma biblioteca para Java chamada lambdaj, que nos permitiria então fazer uso de alguns aspectos de programação funcional [1]. No entanto, esta biblioteca não era satisfatória nos recursos que disponiblizava, e seu uso foi descartado. Ao invés disso, optamos por realizar implementações próprias simulando estas funções.

Optamos por utilizar a API do Android de versão 2.1, pois um dos integrantes do grupo possui um celular com esta versão do sistema operacional instalado, logo pudemos analisar os resultados da aplicação em condições ‘reais’ de utilização e funcionamento.

1.3 Ferramentas de desenvolvimento

Para realizar a codificação, decidimos utilizar a IDE Eclipse, com a qual já estávamos familiarizados, e que nos permitiu estruturar facilmente o projeto do jogo. Além disso, o Eclipse possui um interação muito boa com a SDK do Android, bem como possui suporte para o AVD, Android Virtual Device, que é uma máquina virtual que simula um dispositivo com o sistema operacional Android instalado [2]. Esta máquina virtual foi utilizada durante a criação do projeto para realizar testes sem a necessidade de um dispositivo de telefonia celular com sistema Android.

A linguagem utilizada, Java 7, foi escolhida porque é considerada a linguagem padrão para desenvolvimento de aplicativos Android, e o paradigma orientado a objetos traz uma série de facilidades, como herança, encapsulamento e polimorfismo, o que está de acordo com algumas das especificações deste trabalho.

No entanto, Java não possui suporte nativo a aspectos de programação funcional, o que nos criou a já citada necessidade de simular estes aspectos, o que, na prática, não é eficiente.

Cabe ressaltar que o uso de Java também nos trouxe a vantagem de poder usar o SDK Android, o que nos facilitou grandemente a criação desta aplicação, mas, mesmo assim, sendo nosso primeiro contato com desenvolvimento para Android, demoramos para aprender como dominar o uso dos recursos oferecidos.

2 Aspectos da implementação

2.1 Classes, atributos e métodos

Para a criação do jogo, utilizamos diversas classes, tanto prontas como criadas por nós, cada uma com diversos atributos e métodos. Todas as classes possuem métodos e atributos próprios. Algumas dessas classes incluem objetos gráficos, oriundos de outras classes, que possuem atributos tais como o arquivo de imagem que deve ser carregado e velocidade de deslocamento na tela. Para mudar a velocidade das frutas, por exemplo, utilizamos um método específico que recebe o número da fase atual, e então informa qual a velocidade que deve ser usada para a criação destas frutas.

2.2 Encapsulamentos

Para cada tela do jogo, utilizamos uma interface diferente, cujas implementações foram realizadas em separado. Além disso, utilizamos métodos com todos diferentes níveis de proteção, public, private e protected.

Os métodos public foram escolhidos para possibilitarem acesso global a determinados métodos de uma classe, ou simplesmente para oferecer alguma facilidade de acesso na aplicação. Os métodos private, foram assim escolhidos, pois não se queria que outros objetos ou métodos externos a uma classe tivessem acesso a eles. Já os métodos protected foram escolhidos em classes pais para que estes métodos e atributos pudessem ser passados aos seus descendentes.

2.3 Herança

Nosso jogo possui, basicamente, duas classes distintas de gráficos: a do morcego e a das frutas. Estas duas classes possuem atributos semelhantes, como arquivo de imagem, e métodos semelhantes, como a função de criação na tela. Assim, criamos uma classe, chamada “Model”, da qual tanto a classe “Bat” como “Food” herdam propriedades.

2.4 Composição

Há um uso simples de composição, dentro da classe “Food”, em que criamos um objeto da classe “Speed”, que define justamente como a velocidade deve ser ajustada para a fase atual.

2.5 Polimorfismo por inclusão e paramétrico

Os aspectos de polimorfismo foram pouco explorados no trabalho. Acabamos utilizando apenas sobrecarga de métodos prontos, não aplicamos sobre nenhum método criado por nós mesmos. No caso do polimorfismo paramétrico, só há um exemplo pouco significativo de uso no método de currying.

2.6 Funções de ordem maior e 1ª ordem

Utilizamos a simulação da função de alta ordem map numa classe de debug, que recebe diferentes funções de teste e aplica sobre listas de elementos para nos ajudar a verificar se há algum erro de instanciação ou atribuição. A própria simulação da função map utiliza funções como elementos de primeira ordem.

2.7 Manipulação de listas por funções recursivas

Também na classe com propósitos de debug, fizemos uma função que simula este aspecto. A função percorre os elementos das listas que queremos verificar imprimindo os valores presentes nos elementos. Para tanto, ela realiza uma chamada para ela mesma utilizando como parâmetro o resto (tail) da lista, obtido por uma função também simulada, que segue até que a lista seja vazia.

2.8 Currying

Uma função de currying é simulada na classe de geração das fases. É utilizada aqui pois existem dois parâmetros que devem ser modificados para definir o número de frutas que deve ser comido para que se passe de fase [3].

2.9 Casamento de padrões

Simulamos este aspecto utilizando dois comandos switch/case. Um verifica o valor retornado por uma função randômica e define a imagem utilizada para a fruta a ser criada na tela. O outro também faz a escolha do case com base em um inteiro gerado a partir de uma função randômica. O objeto deste segundo switch/case é escolher se a fruta será criada verticalmente ou horizontalmente, ou seja, determina qual será a trajetória de movimento da fruta.

3 Avaliação da linguagem Java

3.1 Análise

A partir da experiência que tivemos com a linguagem durante o desenvolvimento, consideramos que a linguagem é muito boa para os aspectos de orientação a objeto, e que em termos de reusabilidade e portabilidade é uma das melhores linguagens. A falta de aspectos funcionais, em contrapartida, é um ponto fraco, em especial para a realização de funções recursivas para iterar e a falta de funções de alta ordem. É uma linguagem com grande poder de expressividade, fácil de ler e escrever, mas não é muito ortogonal.

3.2 Tabela de avaliação

As notas são de 1 a 5, com 1 sendo a pior e 5 a melhor.

Característica	Nota	Justificativa
Simplicidade	4	Java possui uma quantidade grande de comandos, mas são fáceis de ler e entender.
Ortogonalidade	2	A linguagem possui um **grande** número de primitivas, e suas combinações são complicadas.
Estruturas de controle	4	Para o paradigma desejado, as estruturas são satisfatórias, mas não há estrutura de controle com recursão.
Tipos e estruturas de dados	5	Grande variedade com potencial enorme de reaproveitamento. Um dos melhores aspectos da linguagem.
Suporte a abstração	4	Bom suporte, mas não fizemos uso de toda essa capacidade.
Expressividade	4	A linguagem é bem expressiva e facilita o uso de seus operadores, mas os operadores para listas e coleções exigem cuidados quando são alterados.
Checagem de tipos	4	A checagem de tipos é bem feita mas pouco flexível, o que evita vários erros, mas também evita usos legítimos da mudança de tipos implícita.
Suporte ao tratamento de exceções	5	O tratador de exceções do Java é muito completo e um grande auxílio para programar e depurar.
Restrições de aliasing	5	Na verdade tivemos problemas com isso, pois nosso programa

		estava alterando uma coleção de objetos e gerava uma exceção. Logo, Java lida bem com múltiplas referências a uma mesma zona memória.
Portabilidade	5	Funciona em qualquer OS e em dispositivos portáteis.
Reusabilidade	5	A orientação a objetos permite que o código seja facilmente modularizado e reutilizado.
Suporte e documentação	5	A documentação é muito rica e a linguagem é sempre atualizada.
Tamanho de código	3	Alguns casos para os quais a linguagem não foi planejada exigem uma quantidade de código grande para contornar a falta de suporte às características destes casos.
Generalidade	5	Java pode ser utilizado em diversas plataformas e para diversos fins, tanto para desktop como para mobiles e sistemas embarcados, além da possibilidade de projetos em web.

Conclusões

Com este trabalho pudemos exercitar muitos dos conceitos propostos pela disciplina, bem como aprender uma plataforma de desenvolvimento, até então desconhecida por ambos os componentes do grupo. Tivemos dificuldades em configurar a SDK do Android com a IDE Eclipse, porém a partir daí o desenvolvimento teve uma boa fluidez e conseguimos um bom resultado.

Também pudemos exercitar um pouco mais da linguagem de programação Java, que não é de domínio do grupo, mas isso não foi empecilho para lidar com a parte funcional descrita na definição do trabalho, outra dificuldade enfrentada.

Referências

[1] Projeto lambdaj. Disponível em: <http://code.google.com/p/lambdaj/>. Acesso em: 17 de junho de 2012.

[2] SDK Android. Disponível em: <http://developer.android.com/sdk/index.html>. Acesso em: 02 de junho de 2012.

[3] Implementação de currying em Java. Disponível em: <http://stackoverflow.com/questions/6134278/does-java-support-currying>. Acesso em: 20 de junho de 2012.