Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenharia Informática & Licenciatura em Informática de Gestão
Programação 1

Trabalho Prático de Avaliação 3
Pong

Beja, 30 de Novembro de 2008

Antes de realizar este trabalho prático deve completar o Guia Prático 6 e (de preferência) também o Guia Prático 7.

Deve verificar cuidadosamente se cumpre integralmente cada um dos requisitos. O cumprimento parcial de um qualquer requisito implica a sua avaliação como não cumprido. Assim, é mais importante cumprir totalmente alguns do que parcialmente todos. Na verdade, cumprir parcialmente todos equivale a não cumprir nenhum.

IMPORTANTE: se não utilizar a linguagem Java™ ou se não utilizar a biblioteca ACM apresentada nas aulas o trabalho será avaliado com zero valores.

1 Introdução

O objectivo geral deste trabalho é o de construir uma versão do jogo Pong, porventura o mais clássico dos jogos clássicos. Veja por exemplo, a entrada na wikipedia em `http://en.wikipedia.org/wiki/Pong`. Aí encontra *links* para muitos sítios com (ainda) mais informação. Mas para ver nada como um vídeo. Por exemplo, este no youtube: `http://www.youtube.com/watch?v=ofgNahDi2hw`. O que se pretende fazer é algo idêntico embora com melhor aspecto, claro. Aqui fica também uma versão mais sofisticada do que a requerida mas que pode dar ideias para eventuais extras: `http://www.miniclip.com/games/battle-pong/en/`.

2 Informações adicionais

Esta secção explica como resolver os problemas fundamentais do jogo, nomeadamente o controlo das raquetes e as colisões da bola. No entanto, não precisa de se preocupar. Como verá, ainda sobra muito para descobrir e fazer!

2.1 Movimento das Raquetes

Para movimentar as raquetes deverá utilizar as teclas. A raquete é mais um objecto da classe `GRect` ou de outra classe que seja um `GObject` (note que um `GCompound` também é um `GObject`). No entanto, terá de existir forma de a mover utilizando as

teclas. Tal é muito simples e muito semelhante ao que temos feito para o rato. O código na Listagem 1 exemplifica como tratar os eventos gerados ao premir (`keyPressed`) e ao libertar (`keyReleased`) uma das teclas `Q`, `A`, seta para cima e seta para baixo. Note que necessita de chamar o método `addKeyListeners()`: `this.addKeyListeners();` para que estes métodos sejam chamados.

Listagem 1: Métodos para tratar o premir e o libertar de uma tecla

```
148   /**
       * Handles event generated wheh a key is pressed
150    * @param e the KeyEvent to handle
       */
152   public void keyPressed(KeyEvent e) // chamado quando é premida uma tecla
      {
154      int kc = e.getKeyCode();
         switch (kc)
156      {
            case KeyEvent.VK_Q:
158            this.leftUp = true;
               break;
160         case KeyEvent.VK_A:
               this.leftDown = true;
162            break;
            case KeyEvent.VK_UP:
164            this.rightUp = true;
               break;
166         case KeyEvent.VK_DOWN:
               this.rightDown = true;
168            break;
         }
170   }

172   /**
       * Handles event generated wheh a key is released
174    * @param e the KeyEvent to handle
       */
176   public void keyReleased(KeyEvent e) // chamado quando é libertada uma tecla
      {
178      int kc = e.getKeyCode();
         switch (kc)
180      {
            case KeyEvent.VK_Q:
182            this.leftUp = false;
               break;
184         case KeyEvent.VK_A:
               this.leftDown = false;
186            break;
            case KeyEvent.VK_UP:
188            this.rightUp = false;
               break;
190         case KeyEvent.VK_DOWN:
               this.rightDown = false;
192            break;
         }
194   }
```

Os métodos na Listagem 1 utilizam atributos (variáveis de instância) do tipo **`boolean`** para assinalar que teclas estão premidas. Para mover as raquetes basta então testar quais as teclas que estão premidas e efectuar o movimento correspondente. Tal está exemplificado na Listagem 2.

Listagem 2: Método para mover as raquetes com base nas teclas premidas e ainda não libertadas

```
      /**
84     * Moves rackets according to key pressed and not yet released
       */
```

```
86    private void moveRacket()
      {
88       if (leftUp)
         {
90          this.left.move(0, -2);
         }
92       if (leftDown)
         {
94          this.left.move(0, 2);
         }
96       if (rightUp)
         {
98          this.right.move(0, -2);
         }
100      if (rightDown)
         {
102         this.right.move(0, 2);
         }
104      this.pause(1);
      }
```

A secção seguinte trata da detecção das colisões da bola com as raquetes.

3 Colisões da bola

A bola pode colidir com as "paredes" ou com as raquetes. As colisões com as paredes podem ser tratadas como já temos feito. A bola move-se da forma usual que já vimos nas aulas (Guias Práticos), ou seja, em linha recta até chocar com uma parede. Nessa altura, faz tabela invertendo o sentido do movimento vertical (quando bate na parede de cima ou de baixo), ou o sentido do movimento horizontal (quando bate numa parede lateral). Quando o joga começa, a bola parte do meio do ecrã (entre as raquetes) num ângulo aleatório. Para obter um ângulo aleatório pode utilizar, por exemplo, o código na Listagem 3 em que `dx` é o deslocamento horizontal e `dy` o vertical.

Listagem 3: Deslocamento da bola

```
52       RandomGenerator gen = RandomGenerator.getInstance();
         this.bdx = this.bdy = 1;
54       if (gen.nextBoolean()) // metade das vezes (em média) troca o sinal
         {
56          this.bdx = -this.bdx;
         }
58       if (gen.nextBoolean()) // metade das vezes (em média) troca o sinal
         {
60          this.bdy = -this.bdy;
         }
```

Para a colisão com as raquetes vamos utilizar o seguinte método da classe `GraphicsProgram`:

```
public GObject getElementAt(double x, double y)
```

Ou seja, fazendo, por exemplo, `GObject obj = this.getElementAt(x, y)` obtemos um de dois possíveis valores em `obj`:

1. **`null`** se não está nenhum objecto no ponto (x, y);

2. a referência para o objecto em (x, y) se lá se encontra um objecto. Se existir mais do que um objecto no ponto (x, y), o método devolve o objecto que está por cima.

Mas como é que utilizamos este método para detectar colisões?

Se utilizarmos um ponto dentro da bola (por exemplo o centro) tal não é muito útil porque o método `getElementAt` iria sempre devolver a bola! Ora nós queremos saber se a bola bateu noutro objecto. É aqui que entra o truque: vamos testar se existem objectos nalguns pontos fora da bola! Estes pontos têm de ser bem escolhidos. A Fig. 1 ilustra as coordenadas desses pontos que correspondem aos cantos do rectângulo mínimo que inclui a bola. Se algum objecto estiver nestes pontos

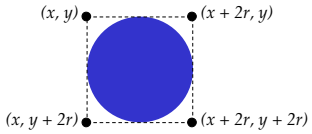

Figura 1: Pontos para testar colisões da bola

então é porque a bola colidiu com esse objecto. Assim, de cada vez que a bola vai para uma nova posição, basta testarmos para cada um destes pontos se lá está um objecto. Podemos então seguir o seguinte algoritmo, a definir num método dentro da classe que tem o método `run`:

1. Chamar o método `getElementAt` para cada um dos pontos, guardando em variáveis booleanas se é ou não igual a **`null`**, ou seja se colidiu com um objecto nesse ponto.
2. Utilizar essas variáveis de forma a inverter o movimento da bola em x, y ou ambos, conforme o que for mais adequado ao movimento pretendido.

Agora já tem toda a informação necessária para realizar a sua versão do Pong!

4 Requisitos Essenciais

Req. 1 - Raquetes Devem existir duas raquetes que se movimentam independentemente, ou seja, uma não afecta o movimento da outra. As raquetes não podem sair da área de jogo. Devem ser utilizadas duas teclas para cada uma das raquetes. Uma das teclas move a raquete para cima; a outra tecla move a raquete para baixo. Assim, necessitará de quatro teclas.

Req. 2 - Bola Deve existir uma bola que surge no meio do ecrã e que seguidamente se movimenta, num sentido aleatório, entre as duas raquetes fazendo ricochete nas raquetes e nas "paredes" de cima e de baixo.

Req. 3 - Pontuação A pontuação de cada jogador deve estar sempre na parte superior da respectiva zona de jogo. Deve utilizar uma *label* para cada jogador. A pontuação processa-se do seguinte modo:

1. No início de cada jogo, a pontuação de ambos os jogadores está a zero.
2. Quando a bola atinge a parede vertical mais próxima de uma raquete, o jogador que controla a outra raquete ganha um ponto.

Req. 4 - Fim do jogo O jogo termina quando um dos jogadores alcança uma quantidade de pontos que é perguntada no início do jogo. Por exemplo: "Pontos necessários para o jogo terminar: ".

Req. 5 - Raquetes originais As raquetes devem ser objectos de uma classe que herda da classe `GCompound`. Por exemplo, pode ser um objecto composto por mais do que um rectângulo, ou por um rectângulo e várias linhas. Veja também o requisito não essencial 8.

Req. 6 - Decomposição em métodos No programa não pode existir nenhum método com mais de 30 linhas. Na contagem excluem-se os comentários e linhas em branco e consideram-se linhas devidamente formatadas. Métodos com instruções **`switch`** ou **`if`** ... **`elseif`**... longas podem, excepcionalmente, conter 60 linhas. Mas, tente evitar esta possibilidade...

5 Requisitos Não Essenciais

Req. 7 - Pontuação máxima O jogo mantém a pontuação máxima alcançada até à data, de entre todas as pontuações já registadas. Esta pontuação máxima é mostrada no topo da área de jogo (entre as pontuações dos jogadores em campo).

Req. 8 - Escolha das raquetes e bola Ambos os jogadores podem escolher de entre dois ou mais objectos diferente para fazerem de raquetes e bola. Estes objectos devem ser instâncias de classes que herdam da classe `GCompound`. Ou seja, é possível escolher qual o objecto pretendido para as raquetes e também, outro diferente, para a bola. Estes objectos podem ser os mesmos do requisito 9.

Req. 9 - Estética do campo O campo deve ter uma linha ao meio e linhas em volta que delimitam o campo e que correspondem às paredes. Também deve ser possível escolher o fundo para o campo de entre um conjunto mínimo de três. O fundo do campo deve corresponder a um objecto, com as dimensões do campo de jogo, de uma classe que herde da classe `GCompound`. Um objecto composto por um padrão de diamantes de uma cor indicada pelo utilizador é um exemplo de um possível fundo. Estes objectos podem ser os mesmos do requisito 8.

Req. 10 - Comportamento da bola Garanta que a bola se comporta de forma natural. Por exemplo, a bola não deve ficar "presa" na raquete ou nas paredes. Mais especificamente, a bola deve sempre fazer ricochetes simples nas raquetes e bola, batendo e mudando logo de direcção, afastando-se da raquete ou parede.

Req. 11 - Diagrama de classes Deve entregar um diagrama de classes que mostre as relações entre as suas classes e entre estas e as classes da biblioteca ACM. Leia o texto sobre diagramas de classes (disponível na página da unidade curricular em `http://kirk.estig.ipbeja.pt/ei/mod/resource/view.php?id=11790`) de forma a que o seu diagrama de classes fique correcto.

Req. 12 - Regras de estilo e elegância do código O código entregue deve respeitar as regras de estilo, nomeadamente **todas** as seguintes:

Apenas os atributos necessários Deve evitar atributos desnecessários. A regra é utilizar variáveis locais, passando-as por parâmetro quando necessário. Como atributos deve colocar apenas os objectos que fazem mesmo parte do jogo, e as variáveis que são utilizadas para comunicar

entre os métodos de tratamento de eventos e os que foram chamados, directa ou indirectamente, pelo método `run`.

Comentários Antes da classe e antes de cada método deve escrever um comentário javadoc(`/** */`) que explique o que o método faz, os respectivos parâmetros e valor devolvido (se existentes). Os comentários podem estar em português mas tente colocá-los em inglês. Os comentários têm de incluir as *tags* `@return` e `@param` para cada parâmetro.

Identificadores em inglês Os nomes de todas as variáveis, métodos e classes devem estar em inglês.

Nomes das variáveis, constantes e classe Utilização de letras minúsculas/-maiúsculas e informação transmitida pelos nomes;

Alinhamento das chavetas Cada uma por baixo da correspondente.

Os espaçamentos Antes e depois dos operadores e das vírgulas.

Indentação coerente e para cada bloco.

Utilização do `this` Utilização da referência **`this`** antes do nome das operações que se aplicam ao objecto da janela de desenho (por exemplo, **`this`**`.add(oneRectangle)`.

Req. 13 - Auto-avaliação No ficheiro .zip entregue deve conter um ficheiro de texto com o nome "auto-aval.txt"que indica quais os requisitos que estão **totalmente** cumpridos (os únicos que contam como cumpridos) e a classificação resultante.

6 Regras para a Realização e Entrega

Este trabalho será avaliado, de acordo com o previsto no Guia da Unidade Curricular (`http://kirk.estig.ipbeja.pt/ei/mod/resource/view.php?id=10101`), em função da quantidade de requisitos essenciais e não essenciais que estejam **totalmente** cumpridos. **Para ser considerado como entregue cada trabalho tem de cumprir todos os seguintes requisitos:**

Quantidade de autores Os trabalhos a entregar têm de ser realizados por grupos de 2 alunos ou por um único aluno.

O que é entregue O trabalho entregue tem de ser um projecto BlueJ pronto a funcionar, sob a forma de um único ficheiro zip contendo toda a directoria do projecto. Antes de entregar, verifique que sabe pôr a funcionar o código no ficheiros zip entregue. Para tal parta desse ficheiro, descompacte-o e tente executá-lo no BlueJ. Tal poderá ser requerido na apresentação individual do trabalho. Se não conseguir pôr a funcionar o conteúdo do ficheiro zip entregue (no moodle e por e-mail) a classificação no trabalho será de zero valores.

Nome do projecto O nome do projecto em BlueJ (abaixo referido como `nomeDoProjecto`) tem de respeitar um dos seguintes formatos, conforme o trabalho tenha sido realizado por um só aluno ou por dois alunos. Note que `Primeiro1` e `Ultimo1` representam o primeiro e o último nome de um autor, e `Primeiro2` e `Ultimo2` representam o primeiro e o último nome do outro autor. `Numero1` e `Numero2` representam os números de aluno de cada autor.

Para um autor:

```
Numero1_Primeiro1Ultimo1_TP3_P1_2008-2009
Por exemplo: 1232_AnaGomes_TP3_P1_2008-2009
```

Para dois autores:

```
Numero1_Primeiro1Ultimo1_Numero2_Primeiro2Ultimo2_TP3_P1_2008-2009
Por exemplo: 1232_AnaGomes_3454_JoaoSilva_TP3_P1_2008-2009
```

O trabalho é entregue compactando a directoria do projecto BlueJ num ficheiro .zip de forma a que este fique com o nome `nomeDoProjecto.zip`.

Entrega A entrega tem de ser feita num ficheiro no formato zip com o nome `nomeDoProjecto.zip` e por duas vias:

1. Na página da disciplina. Os trabalhos realizados por grupos de dois alunos devem ser entregues têm de ser entregues via moodle por **um** dos elementos do grupo.
2. Por e-mail, respeitando as seguintes regras: A entrega por e-mail é feita para `trabalhos.p1ARROBAgmail.com`. O e-mail a enviar deve conter em attach um único ficheiro no formato zip. O subject do e-mail deve ser o `nomeDoProjecto`. Pode entregar mais do que uma vez, desde que dentro do prazo. A última entrega dentro do prazo é a única que conta.

Data limite de entrega O trabalho tem de ser entregue no moodle e por e-mail até às **14:00 de 16 de Dezembro de 2008**. **Os trabalhos que não tenham sido entregues até essa data e hora, nem no moodle nem por e-mail, serão considerados como não entregues e avaliados com zero valores**.

Finalmente, note que necessita de realizar mais do que o pedido para obter mais de 16 valores. Aqui ficam algumas sugestões, mas note que a criatividade também é sempre necessária para alcançar a nota máxima. A implementação correcta e perfeita de dois dos pontos seguinte garante mais de 16 valores mas só se todos os 13 requisitos estiverem perfeitamente cumpridos.

- Possibilidade de cada jogador controlar duas raquetes: uma mais à frente e outra mais atrás.
- Possibilidade escolher as teclas, bem como de jogar com duas bolas em simultâneo.
- Surgimento de objectos móveis malignos que quando atingem a raquete fazem perder pontos.
- Ricochete de ângulos diferentes conforme o sítio da raquete que é atingido.

Procure também as versões do Pong disponíveis na Internet para obter mais ideias!

A criatividade também pode justificar uma melhor classificação pelo que extras mais originais e sofisticados serão uma boa aposta. Note que estas estas adições só contam para a classificação do trabalho se forem consideradas suficientemente significativas e se **todos** os requisitos estiverem completamente cumpridos.

7 Nota importante

Todas as contribuições para o trabalho que não sejam da exclusiva responsabilidade dos autores têm de ser identificadas (com comentários no código e referências no relatório) e a sua

utilização bem justificada e expressamente autorizada pelo professor responsável. Justificações insatisfatórias, ausência de autorização, ou ausência de referências para trabalhos ou colaborações externas utilizadas serão consideradas fraude sempre que os trabalhos sejam considerados demasiado semelhantes para terem sido criados de forma independente. **Tal terá como consequência a reprovação na unidade curricular de todos os alunos envolvidos**. Excepcionalmente, poderão ser penalizados apenas os alunos que se declarem como únicos culpados. Assim, nenhum aluno deve dar cópia do seu código (ainda que em fase inicial) a outro. Por essa mesma razão não é boa ideia partilhar código com os colegas, quer directamente quer através do fórum. Naturalmente, cada aluno pode trocar impressões e esclarecer dúvidas com todos os colegas, mas deve escrever o seu código apenas com a colaboração do seu colega de grupo. Deve também saber escrever todo o código sozinho. Lembre-se que será avaliado num teste prático final em frente a um computador. A classificação neste trabalho prático fica dependente de uma apresentação individual do mesmo, tal como previsto no guia da unidade curricular.

Finalmente, leia com MUITA atenção todo o enunciado. A falha de parte do exigido num requisito implica o não cumprimento desse requisito e consequente penalização.

Bom trabalho!

João Paulo Barros