Mais dois exercícios de Processing

Aquário

Os *screensavers* com peixinhos são muito populares. Mas não nos ficaria bem usar *screensavers* com peixinhos, comprados ou sacados. Programador que se preza produz os seus próprios *screensavers*.

Pois bem, escrevamos um programa Processing que funcione como um *screensaver* de peixes num aquário. Haverá peixes de várias qualidades, uns andando da esquerda para a direita e outros da direita para a esquerda, sempre na horizontal, cada um à sua velocidade. Quando um peixe sai do computador pelo lado direito volta a entrar pelo lado esquerdo, e vice-versa.

Requisitos técnicos.

O programa deve ter uma classe `Fish`, para representar peixes. Esta classe deve ter membros para descrever a posição do peixe, a velocidade e a cor, e deve ter um método `draw`, sem argumentos, para desenhar o peixe.

Os peixes do aquário devem estar todos num array `fishes`, assim declarado:

```
Fish fishes[];
```

Liberdade artística

Todos temos liberdade artística para interpretar criativamente este texto, respeitando sempre os requisitos técnicos.

Nível 1

Considere apenas dois peixes, com cores diferentes e velocidades diferentes, um movendo-se da esquerda para a direita e outro movendo-se da direita para a esquerda. Cada peixe é desenhado esquematicamente na forma de um retângulo.

Nível 2

Como o anterior, mas com quatro ou mais peixes, agora com forma de peixe.

O método `draw` deve usar uma combinação de formas geométricas que dê ao desenho um aspeto de peixe, visto de lado. Por exemplo, devemos ver claramente a barbatana caudal, talvez um triângulo, a boca do peixe, talvez um segmento, e o olho do peixe, talvez um ponto "gordo". O desenho deve ser concebido de forma a poder se ampliado ou encolhido sem perder a forma. Assim poderemos ter peixes mais redondos (estilo peixe-lua) ou mais compridos (estilo peixe-espada), ou intermédios (estilo dourada).

Os peixes no aquário devem ser de tamanhos variados e diversas cores.

Nível 3

Acrescente ao anterior bolhas de ar a subir do fundo para a superfície, a velocidade constante. Bolhas maiores sobem mais devagar (porque fazem mais atrito na água).

Nível 4

Defina uma classe `Sardine` para representar sardinhas. As sardinhas são um peixe pequeno, prateado. Represente depois um cardume de pelo menos 16 sardinhas, deslocando-se em formação, todas à mesma velocidade. As posições das sardinhas no cardume são fixadas aleatoriamente dentro de um círculo à escolha. Isto é, o cardume dever ter aproximadamente a forma de um círculo.

Fisga

Este programa é um jogo de caçar pardais à fisga, no computador.

Do lado esquerdo da janela há uma fisga; do lado direito há uma árvore onde estão pousados vários pardais. Queremos acertar nos pardais com a fisga.

Para acertar nos pardais, podemos orientar a fisga e controlar a velocidade inicial da pedra. Temos de contar com a força da gravidade, mas ignoramos o atrito da pedra no ar.

Inicialmente a fisga tem a pedra lá dentro. Estão ambas paradas e nada acontece. Carregando com o rato algures na janela, deve surgir um segmento entre o rato e a fisga. O sentido do segmento (do rato para a fisga) representa o sentido inicial da pedra, ao sair da fisga. O comprimento do segmento é proporcional à velocidade inicial da fisga: com um segmento menor, a pedra sai mais devagar, com um segmento maior, sai mais depressa.

Requisitos técnicos.

O programa deve ter uma classe `Slingshot`, para agrupar todas as variáveis que caraterizam o comportamento da fisga, incluindo a pedra.

Nos níveis 3 e 4 deve haver também uma classe `Bird` para representar cada um dos pardais na árvore. O bando será representado por um array, assim declarado:

```
Bird birds[];
```

Nível 1

Programar uma fisga sem gravidade.

A pedra deve sair da fisga em linha reta e velocidade constante, com a tecla 'S' (de "*shoot*"). A trajetória da fisga deve ser desenhada, em tempo real. A tecla 'R' (de "*restart*") interrompe a trajetória e volta ao estado inicial. Calibre as constantes do programa de maneira que o movimento da pedra seja suave e percetível.

Nível 2

Acrescentar a gravidade ao caso anterior.

A trajetória deve agora ser uma parábola. Arbitre uma constante de gravidade que seja adequada à nossa simulação.

Nível 3

Realizar o jogo.

Incluir esquematicamente os pardais na árvore. Quando a pedra acerta no pardal, o pardal cai da árvore e desaparece (não é preciso mostrar o pardal a cair…) e o sistema recomeça, com menos esse pardal. Se o tiro se perder, saindo da janela pela direita ou por baixo, o sistema recomeça, com os mesmos pardais.

Nível 4

Implementar uma fisga computacional.

Dada a posição de um pardal, a fisga calcula a velocidade inicial e ângulo de tiro de maneira a acertar sempre, de cima para baixo.

A fisga escolhe um dos pardais, prepara o tiro, sozinha, e só temos de disparar, com 'S'. Nunca falha. Note que a pedra deve acertar no pardal no percurso descendente.

Nível extra-concurso

Enfeite o programa a seu gosto com bonequinhos esquemáticos para os pardais, para a árvore e para a fisga. Mostre aos seus pais. Ficarão orgulhosos. Foi para isto mesmo que o mandaram tirar engenharia informática. 🤓