PC Robots

V. 1.4 V.Lobo Escola Naval 2006

Introdução

Simulador PC Robots

- Simulador de robots desenhado por P.D.Smith
  - Versão 1.41, P.D.Smith, 1992
- Objectivo nesta cadeira
  - Tomar contacto com a programação de sistemas em tempo real, e sistemas de robótica
  - Adquirir prática de programação em C
- Objectivo desta exposição
  - Facilitar a leitura da documentação do simulador
  - Dar algumas sugestões para a elaboração do programa

1

O Robot

Simulador PC Robots

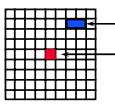
- Cada robot dispõe de:
  - Um motor que lhe permite movimentar-se em qualquer direcção (função movement)
  - Um sistema de navegação que fornece as suas coordenadas(função Getxy)
  - Um canhão que dispara projecteis a uma determinada distância numa determinada direcção
  - Uma bateria que fornece energia ao motor, e ao canhão, e que dispõe de mecanismos de carregamento de energia
  - Um radar de ataque (função Scan)
  - Um radar de navegação (função Get_local_map)
- Cada robot é composto por um programa executável
  - Esse programa deverá chamar rotinas de baixo nível que fazem o robot interagir com o ambiente
  - As rotinas de baixo nível são executadas pelo simulador (e não por um robot real!)

2

A arena

Simulador PC Robots

- Os robots movem-se numa arena com 1000x1000 pixels.
- A arena pode estar vazia ou pode ter obstáculos (paredes, pontos de recarga, fossos), dependendo do ficheiro “pcrobots.rna”
- Cada conjunto de 10x10 pixels forma um “caracter”, que formam um outro sistema de coordenadas (em que a arena tem 100x100).
  - A rotina Get_local_map devolve uma matriz de caracteres:



3

Rotinas mais importantes

Simulador PC Robots

- Configure
  - Permite determinar as capacidades do robot. Cada robot tem 10 pontos que têm que ser divididos entre as várias características: velocidade, resistência, poder de fogo, etc.
- Movement
  - Põe o robot em movimento numa determinada direção.
  - Os movimentos e as acelerações não são instantâneas.
- Scan
  - Se existir um outro robot na direcção pretendida, devolve o seu “track number” e a distância.
- Shoot
  - Dispara na direcção indicada, para a distância indicada.
  - O canhão demora algum tempo até poder voltar a disparar

4

Exemplo (1 e 2)

Simulador PC Robots

```
#include "pcrobots.h"
#include "pcrobots.c"

main()
{
 configure(2,2,2,2,2,0);
 while(1)
    movement( 2, 0 );
}
```

O robot irá mover-se lentamente (v=2) para o azimute 0. Acabará por se destruir contra a parede da extremidade da arena

```
#include "pcrobots.h"
main()
{
 int x,y,
     direction;

 configure(2,2,2,2,2,0); /* Configura o robot */
 direction=0;            /* Direccao inicial = 0 */

 while(1)                /* Ciclo infinito principal */
 {
  getxy( &x, &y );       /* Obter as coordenadas do robot */
                         /* Se esta para bater no lado direito... */
  if( (direction==0) && (x>950) )
            direction=180;       /* direccao passa a ser 180 */
  else if( (direction==180) && (x<50) )
            direction=0;

  movement( 20, direction );
 }
}
```

5

Exemplo (3)

Simulador PC Robots

```
#include "pcrobots.h"

main()
{
 int x,y,
          direction,
          ang,dist;

 configure(2,2,2,2,2,0);          /* Configura o robot */
 direction=0;                     /* Direccao inicial = 0 */

 while(1)                         /* Ciclo infinito principal */
 {
  getxy( &x, &y );                /* Obter as coordenadas do robot */

  direction = Get_new_direction( x,y,direction );
  Get_target(&ang,&dist);
  shoot(ang,dist);

  movement( 20, direction );
 }
}
```

6

Exemplo (cont.3)

Simulador PC Robots

```
/*************************************************************************
* NOME: Get_new_direction
*
* OBJECTIVO: Dada a posicao e a direccao actual decidir a melhor direccao
*                                   a seguir
*
* PARAMETROS: x,y                 : coordenadas da posicao actual
*                                   dir   : direccao
*
* DATA: 12/10/94 by V.Lobo
*************************************************************************/
int Get_new_direction(int x, int y, int dir)
{
 if( (dir==0) && (x>950) )        /* Se esta para bater no lado direito */
   return 180;                    /* Direcao passa a ser 180 */
 if( (dir==180) && (x<50) )       /* Se esta para bater no lado esquerdo*/
   return 0;                      /* Direcao passa a ser 0 */
 return dir;                      /* Senao, mantem a mesma direcao */
 }
```

```
*************************************************************************
* NOME: Get_target
*
* OBJECTIVO: Obter a marcacao e dist. de um alvo
*
* PARAMETROS:
*
* DATA: 12/10/94 by V.Lobo
*************************************************************************/
Get_target(int *marcacao, int *distancia)
{
 int i=0;
 while( scan(i, 5, distancia) == -1 )        /* Enquanto nao encontrar nada...*/
   i+=10;                                    /* Rodar o radar */
 *marcacao=i;
 return 1;
 }
```

7

Trabalho Prático

Simulador PC Robots

- **Robot combatente simples**
  - Construa um robot capaz de combater outros.
  - COMENTE devidamente o código que escrever !
  - Haverá um torneiro entre os robots produzidos ...
- **Robot de reconhecimento**
  - Construa um robot que consiga fazer um mapa da arena onde se encontra.
  - Esse mapa deverá ser uma matriz de 100x100 com caracteres a representar arena livre, muros, fossos, etc.
  - O robot deverá guardar esse mapa num ficheiro, de modo a poder ser analisado após a "morte" do robot
  - Após terminar, o robot pode-se "suicidar" contra uma parede
- **Robot de regata**
  - Construa um robot que dê voltas a um "tirângulo olímpico" com "boias" nas coordenadas (10,10), (10,70), e (70,10). Ganha o primeiro robot que completar 10 voltas, começando e acabando na bóia 1

8

Trabalho Prático (2)

Simulador PC Robots

- **Robot-míssil**
  - Construa um robot que procure um fosso, e se atire para cima desse fosso (como se fosse um míssil sobre um alvo)
- **Robot-escolta**
  - Construa um robot que se mantenha num raio de 15 pixels do robot cujo "track-number" é 1.
- **Robot que usa apontadores**
  - Construir um robot capaz de combater outros.
  - O robot TEM QUE ter pelo menos uma estrutura de dados que use apontadores e criação dinâmica de variáveis. Sugere-se que o robot crie uma lista com os diferentes contactos que vai fazendo, e para cada contacto tenha uma "sub-lista" com as posições e a hora (ticks) a que foram avistados. Essa lista deverá ser periodicamente enviada para o disco (para que fique um registo no caso do robot ser abatido).

9