PRÀCTICA 2

FONAMENTS DE SISTEMES OPERATIUS

THREADS + PROCESSOS

Jan Torres Rodríguez

Raul Martín Morales

2023/24

Index:

1. Fase 1

1.1. Especificacions

A la primera fase de la pràctica caldrà modificar les funciones que s'encarreguen del moviment de les paletes de l'ordinador i de l'usuarí, i també la pilota. D'aquesta manera fer que puguin funcionar aquestes funciones com a fils d'execució independetns.

A la funció mou_paleta_ordinador, que será la que més modificarem, entrarà per prametre un index que indicará quina paleta volem controlar, index = 0 controlarà la primera paleta i index =1 controlarà la segona paleta. I cada paleta s'escriurà com a caràcter invertit amb cod ASCII.

A diferencia, en les funcions moure_pilota i mou_paleta_usuari pasarem el paràmetre cap que no portarà cap informació. Ara aquestes tres funcions de moviment s'executaran de manera independent, i la finalització dels fils de moviment es faran de controlar segons acabi el joc.

El important de la fase es que haurem de implementar que es pugin controlar fins a 9 paletes, i per a cada paleta guardar la seva informació en un vector. Per això haurem de modifcar el carregar_parametres per que es puguin crear 9 paletes.

Per altra banda, s'ha de controlar amb alguna funció que controli el temps del joc i que es mostrar-ho per pantalla.

Per aconseguir un bon funcionament dels threads, s'haurà de crear un thread per a cada una de las paletes existents, i després altres 3 per al moviment de la paleta usuari, moviment pilota i rellotge

Per últim quan es compleixi que s'acaba el joc el fil d'execució principal sortirà del bucle i passarà a esperar a que tots els fils finalitzin la seva execució abans de eliminar-los.

1.2. Disseny

Hem decidit crear diferents arrays per guardar la informacio de les posibles 9 paletes amb un maxim de MAX_PALETES=9. Per altre banda hem creat una variable de tipus enter per guardar el número de paletas que llegim i també un variable temps per gestionar el temps del joc. També creem una variable anomenada tid que sera un array

de tipus pthread_t on guardarem els identificadors de cada thread que farem ús en aquesta pràctica.

Al carrega_parametres, en la part on es crea les paletes de l'ordinador, ara gestionem les diferents paleres que es poden crear amb un bucle on anirem incrementant el número de paletes.

A la funció inicialitza_joc hem afegit diverses línees per poder dibuixar les paletes en les seves posicions inicials.

En totes les funcions on es produeixen moviments hem afegit un bucle per a que es faci la funcio sempre mentre no s'acabi el joc.

A la funcio mou_paleta_ordinador ara li pasem per paràmetre el index de cada paleta i així controlar cada paleta segons el index en el array de la informació de cada paleta.

I hem afegit una funció anomenada rellotge que ens ajudarà a poder anar incrementant el temps i així indicar per pantalla quants minuts i segons porta en la partida.

Per últim al main fem la creació dels threads de les 9 paletas de l'ordinador, i les altres 3 funcions .

1.3. Implementació

Tennis1.c:

Tennis1.c:

```
#include <stdio.h>                      /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include "winsuport.h"                  /* incloure definicions de funcions propies */
#include <time.h>
#include <pthread.h>
#include <stdint.h> /* definició de intptr_t per màquines de 64 bits */
#include <unistd.h>

#define MIN_FIL 7                       /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80
#define MIN_PAL 3
#define MIN_VEL -1.0
#define MAX_VEL 1.0
#define MIN_RET 0.0
#define MAX_RET 5.0
```

```
#define MAX_PALETES 9
#define MAX_THREADS 12

/* variables globals */
pthread_t tid[MAX_THREADS];
float po_pf[MAX_PALETES];                   /* pos. vertical de la paleta de l'ordinador, en valor real */
int ipopf[MAX_PALETES], ipopc[MAX_PALETES];            /* posicio del la paleta de l'ordinador */
float vpal[MAX_PALETES];
float palret[MAX_PALETES];
int paletes;
int temps = 0;

int n_fil, n_col, m_por;        /* dimensions del taulell i porteries */
int l_pal;                                  /* longitud de les paletes */
float v_pal;                                /* velocitat de la paleta del programa */
float pal_ret;                              /* percentatge de retard de la paleta */

int ipu_pf, ipu_pc;             /* posicio del la paleta d'usuari */
int ipo_pf, ipo_pc;             /* posicio del la paleta de l'ordinador */

int ipil_pf, ipil_pc;                       /* posicio de la pilota, en valor enter */
float pil_pf, pil_pc;                       /* posicio de la pilota, en valor real */
float pil_vf, pil_vc;                       /* velocitat de la pilota, en valor real*/
float pil_ret;                              /* percentatge de retard de la pilota */

int retard;                         /* valor del retard de moviment, en mil.lisegons */
int moviments, moviments_inicials;                  /* numero max de moviments paletes per acabar el joc */

int cont, tec;


/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins un fitxer de text, el nom del qual es passa per referencia en   */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-     */
/* tinent al SO (segons comentaris del principi del programa).                 */
void carrega_parametres(const char *nom_fit)
{
  FILE *fit;

  fit = fopen(nom_fit,"rt");                    /* intenta obrir fitxer */
  if (fit == NULL)
```

```
 {                  fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
                    exit(2);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %d %d\n",&n_fil,&n_col,&m_por,&l_pal);
 if ((n_fil < MIN_FIL) || (n_fil > MAX_FIL) ||
(n_col < MIN_COL) || (n_col > MAX_COL) || (m_por < 0) ||
 (m_por > n_fil-3) || (l_pal < MIN_PAL) || (l_pal > n_fil-3))
 {
fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");
fprintf(stderr,"\t%d =< n_fil (%d) =< %d\n",MIN_FIL,n_fil,MAX_FIL);
fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);
fprintf(stderr,"\t0 =< m_por (%d) =< n_fil-3 (%d)\n",m_por,(n_fil-3));
fprintf(stderr,"\t%d =< l_pal (%d) =< n_fil-3 (%d)\n",MIN_PAL,l_pal,(n_fil-3));
fclose(fit);
exit(3);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %f %f %f\n",&ipil_pf,&ipil_pc,&pil_vf,&pil_vc,&pil_ret);
 if ((ipil_pf < 1) || (ipil_pf > n_fil-3) ||
(ipil_pc < 1) || (ipil_pc > n_col-2) ||
(pil_vf < MIN_VEL) || (pil_vf > MAX_VEL) ||
(pil_vc < MIN_VEL) || (pil_vc > MAX_VEL) ||
(pil_ret < MIN_RET) || (pil_ret > MAX_RET))
 {
fprintf(stderr,"Error: parametre pilota incorrectes:\n");
fprintf(stderr,"\t1 =< ipil_pf (%d) =< n_fil-3 (%d)\n",ipil_pf,(n_fil-3));
fprintf(stderr,"\t1 =< ipil_pc (%d) =< n_col-2 (%d)\n",ipil_pc,(n_col-2));
fprintf(stderr,"\t%.1f =< pil_vf (%.1f) =< %.1f\n",MIN_VEL,pil_vf,MAX_VEL);
fprintf(stderr,"\t%.1f =< pil_vc (%.1f) =< %.1f\n",MIN_VEL,pil_vc,MAX_VEL);
fprintf(stderr,"\t%.1f =< pil_ret (%.1f) =< %.1f\n",MIN_RET,pil_ret,MAX_RET);
fclose(fit);
exit(4);
 }

paletes = 0;
while (!feof(fit) && paletes < MAX_PALETES){

 fscanf(fit,"%d %d %f %f\n",&ipo_pf,&ipo_pc,&v_pal,&pal_ret);
 ipopf[paletes] = ipo_pf;
 ipopc[paletes] = ipo_pc;
 vpal[paletes] = v_pal;
```

```
 palret[paletes] = pal_ret;


 if ((ipopf[paletes] < 1) || (ipopf[paletes]+l_pal > n_fil-2) ||
(ipo_pc < 5) || (ipo_pc > n_col-2) ||
(v_pal < MIN_VEL) || (v_pal > MAX_VEL) ||
(pal_ret < MIN_RET) || (pal_ret > MAX_RET))
   {
fprintf(stderr,"Error: parametres paleta ordinador incorrectes:\n");
fprintf(stderr,"\t1 =< ipo_pf (%d) =< n_fil-l_pal-3 (%d)\n",ipo_pf,(n_fil-l_pal-3));
fprintf(stderr,"\t5 =< ipo_pc (%d) =< n_col-2 (%d)\n",ipo_pc,(n_col-2));
fprintf(stderr,"\t%.1f =< v_pal (%.1f) =< %.1f\n",MIN_VEL,v_pal,MAX_VEL);
fprintf(stderr,"\t%.1f =< pal_ret (%.1f) =< %.1f\n",MIN_RET,pal_ret,MAX_RET);
fclose(fit);
exit(5);
   }
   paletes++;
}
 fclose(fit);                                    /* fitxer carregat: tot OK! */


}



/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
int inicialitza_joc(void)
{
 int i, i_port, f_port, retwin;
 char strin[51];


 retwin = win_ini(&n_fil,&n_col,'+',INVERS);   /* intenta crear taulell */


 if (retwin < 0)      /* si no pot crear l'entorn de joc amb les curses */
 { fprintf(stderr,"Error en la creacio del taulell de joc:\t");
   switch (retwin)
   {  case -1: fprintf(stderr,"camp de joc ja creat!\n");
              break;
      case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
break;
      case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
              break;
      case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
              break;
   }
```

```
   return(retwin);
  }

  i_port = n_fil/2 - m_por/2;      /* crea els forats de la porteria */
  if (n_fil%2 == 0) i_port--;
  if (i_port == 0) i_port=1;
  f_port = i_port + m_por -1;
  for (i = i_port; i <= f_port; i++)
  {                    win_escricar(i,0,' ',NO_INV);
win_escricar(i,n_col-1,' ',NO_INV);
  }

  ipu_pf = n_fil/2; ipu_pc = 3;                                    /* inicialitzar pos. paletes */
  if (ipu_pf+l_pal >= n_fil-3) ipu_pf = 1;
  for (i=0; i< l_pal; i++)              /* dibuixar paleta inicialment */
  {
    win_escricar(ipu_pf +i, ipu_pc, '0',INVERS);
  for (int x=0; x<paletes; x++){
     win_escricar(ipopf[x] +i, ipopc[x], '1',INVERS);
     po_pf[x] = ipopf[x];
   }
  }

  pil_pf = ipil_pf; pil_pc = ipil_pc;                /* fixar valor real posicio pilota */
  win_escricar(ipil_pf, ipil_pc, '.',INVERS); /* dibuix inicial pilota */
  return(0);
}



/* funcio per moure la pilota; retorna un valor amb alguna d'aquestes                  */
/* possibilitats:                                                                                                                  */
/*                    -1 ==> la pilota no ha sortit del taulell                                */
/*                     0 ==> la pilota ha sortit per la porteria esquerra                  */
/*                    >0 ==> la pilota ha sortit per la porteria dreta                         */
void * moure_pilota(void * cap)
{
  int f_h, c_h;
  char rh,rv,rd,pd;

do{
  f_h = pil_pf + pil_vf;                              /* posicio hipotetica de la pilota */
  c_h = pil_pc + pil_vc;
```

```
 cont = -1;                     /* inicialment suposem que la pilota no surt */
 rh = rv = rd = pd = ' ';
 if ((f_h != ipil_pf) || (c_h != ipil_pc))
 {                              /* si posicio hipotetica no coincideix amb la pos. actual */
  if (f_h != ipil_pf)           /* provar rebot vertical */
  {
    rv = win_quincar(f_h,ipil_pc);      /* veure si hi ha algun obstacle */
  if (rv != ' ')                /* si no hi ha res */
     {  pil_vf = -pil_vf;       /* canvia velocitat vertical */
     f_h = pil_pf+pil_vf;       /* actualitza posicio hipotetica */
     }
  }
  if (c_h != ipil_pc)           /* provar rebot horitzontal */
  {
    rh = win_quincar(ipil_pf,c_h);      /* veure si hi ha algun obstacle */
     if (rh != ' ')             /* si no hi ha res */
      {   pil_vc = -pil_vc;     /* canvia velocitat horitzontal */
          c_h = pil_pc+pil_vc;  /* actualitza posicio hipotetica */
      }
  }
  if ((f_h != ipil_pf) && (c_h != ipil_pc))     /* provar rebot diagonal */
   {rd = win_quincar(f_h,c_h);
     if (rd != ' ')                             /* si no hi ha obstacle */
     {   pil_vf = -pil_vf; pil_vc = -pil_vc;    /* canvia velocitats */
         f_h = pil_pf+pil_vf;
         c_h = pil_pc+pil_vc;       /* actualitza posicio entera */
     }
  }
  if (win_quincar(f_h,c_h) == ' ')  /* verificar posicio definitiva */
  {                                                             /* si no hi ha obstacle */
win_escricar(ipil_pf,ipil_pc,' ',NO_INV);   /* esborra pilota */
pil_pf += pil_vf; pil_pc += pil_vc;
ipil_pf = f_h; ipil_pc = c_h;       /* actualitza posicio actual */
if ((ipil_pc > 0) && (ipil_pc <= n_col))    /* si no surt */
win_escricar(ipil_pf,ipil_pc,'.',INVERS); /* imprimeix pilota */
else{
   if (ipil_pc > n_col) cont = 0;
   else if(ipil_pc < 0) cont = 1;
 }
  }
 }
 else { pil_pf += pil_vf; pil_pc += pil_vc; }
```

```
 win_retard(retard);

}while ((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));

}

/* funcio per moure la paleta de l'usuari en funcio de la tecla premuda */
void * mou_paleta_usuari(void * cap)
{
 do{
 tec=win_gettec();

 if ((tec == TEC_AVALL) && (win_quincar(ipu_pf+l_pal,ipu_pc) == ' '))
 {
  win_escricar(ipu_pf,ipu_pc,' ',NO_INV);   /* esborra primer bloc */
  ipu_pf++;                                                    /* actualitza posicio */
  win_escricar(ipu_pf+l_pal-1,ipu_pc,'0',INVERS); /* impri. ultim bloc */
  if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
 }
 if ((tec == TEC_AMUNT) && (win_quincar(ipu_pf-1,ipu_pc) == ' '))
 {
  win_escricar(ipu_pf+l_pal-1,ipu_pc,' ',NO_INV); /* esborra ultim bloc */
  ipu_pf--;                                                     /* actualitza posicio */
  win_escricar(ipu_pf,ipu_pc,'0',INVERS);                 /* imprimeix primer bloc */
  if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
 }
 if (tec == TEC_ESPAI){
  do{
   tec = win_gettec();
  }while (tec != TEC_ESPAI);
 }
 win_retard(retard);

 }while((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));
}

/* funcio per moure la paleta de l'ordinador autonomament, en funcio de la */
/* velocitat de la paleta (variable global v_pal) */
void * mou_paleta_ordinador(void * index)
```

```
{
 int f_h[MAX_PALETES];
/* char rh,rv,rd; */

 int i = (intptr_t) index;
 do{
    f_h[i] = po_pf[i] + vpal[i];                    /* posicio hipotetica de la paleta */
    if (f_h[i] != ipopf[i])          /* si pos. hipotetica no coincideix amb pos. actual */
    {
    if (vpal[i] > 0.0)                                          /* verificar moviment cap avall */
    {
  if (win_quincar(f_h[i]+l_pal-1,ipopc[i]) == ' ')   /* si no hi ha obstacle */
  {
                        win_escricar(ipopf[i],ipopc[i],' ',NO_INV);     /* esborra primer bloc */
    po_pf[i] += vpal[i]; ipopf[i] = po_pf[i];                   /* actualitza posicio */
    win_escricar(ipopf[i]+l_pal-1,ipopc[i],'1',INVERS); /* impr. ultim bloc */
    }
  else                                  /* si hi ha obstacle, canvia el sentit del moviment */
    vpal[i] = -vpal[i];
    }
    else                                            /* verificar moviment cap amunt */
    {
    if (win_quincar(f_h[i],ipopc[i]) == ' ')      /* si no hi ha obstacle */
    {
      win_escricar(ipopf[i]+l_pal-1,ipopc[i],' ',NO_INV); /* esbo. ultim bloc */
      po_pf[i] += vpal[i]; ipopf[i] = po_pf[i];                 /* actualitza posicio */
      win_escricar(ipopf[i],ipopc[i],'1',INVERS);               /* impr. primer bloc */
      }
  else                                  /* si hi ha obstacle, canvia el sentit del moviment */
    vpal[i] = -vpal[i];
    }
    }
    else{
      po_pf[i] += vpal[i];          /* actualitza posicio vertical real de la paleta */
    }
    win_retard(retard);

 }while((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));

}

void * rellotge(void * cap) {
```

```
  cont = -1;
  char missatge[100];
  while (1) {
    temps++;
    sleep(1);
    int segs = temps%60;
    int min = temps/60;

    sprintf(missatge, "Movs: %d restants|Temps: %d:%d",
        moviments, min, segs);
    win_escristr(missatge);

    if (moviments == 0 || cont != -1) {
      win_fi();
      if (cont == 1)
        printf("Ha guanyat l'ordinador!\n");
      else if (cont == 0)
        printf("Ha guanyat l'usuari!\n");
      exit(0);
    }
    win_retard(retard);
  }
  return NULL;
}

/* programa principal*/
int main(int n_args, const char *ll_args[])
{
 if ((n_args != 3) && (n_args !=4))
 {          fprintf(stderr,"Comanda: tennis0 fit_param moviments [retard]\n");
            exit(1);
 }
 carrega_parametres(ll_args[1]);
 moviments=atoi(ll_args[2]);

 if (n_args == 4) retard=atoi(ll_args[3]);
 else {
  retard=100;
 }

 if (inicialitza_joc() !=0)   // intenta crear el taulell de joc
  exit(4);              // aborta si hi ha algun problema amb taulell
```

```
 for (int i=0; i<paletes; i++){
   pthread_create(&tid[i], NULL, mou_paleta_ordinador, (void *)(intptr_t)i);
 }
 pthread_create(&tid[paletes], NULL, mou_paleta_usuari, NULL);
 pthread_create(&tid[paletes+1], NULL, moure_pilota, NULL);
 pthread_create(&tid[paletes+2], NULL, rellotge, NULL);

 for (int i=0; i<paletes+3; i++){
   pthread_join(tid[i], NULL);
 }

 win_fi();
 return(0);
}
```

1.4. Joc de proves

Primer joc de proves que hem fet a la fase 1 es comprovar que s'executi el programa amb el fitxer camp1.txt on es crea només una paleta de l'ordinador correctament.

Per altra banda, hem comprovat que el programa funcioni carregant el camp2.txt on es creen 3 paletes per a l'ordinador.

Després hem comprovat que l'altre fitxer que tenim que es el camp3.txt on es generen 5 paletes, que és el màxim, per a l'ordinador correctament.

I l'útim fitxer anomenat camp4.txt carrega ve les 9 paletes i el joc funciona correctament.

També hem hagut de comprovat diferents implementacions que s'ens ha proposat, com que quan es premi la tecla RETURN s'acabi el joc i funciona correctament. I quan finalitza el joc sempre es mostra el temps que ha durat la partida per línea de comanda.

Hem comprovat que quan els moviments s'esgotin el joc acabi i ha funcionat correctament.

Per últim hem verificat que quan l'usuari o l'ordinador guanyi es mostri per pantalla segons qui hagi guanyat.

2. Fase 2

2.1. Especificacions

A la fase 2 s'haurà de fer l'ús de semàfors per sincronitzar els threads. Caldrà indicar seccions crítiques que evitin problemes de concurrència entre elss múltiples fils d'execució que hem creat a la fase 1, s'haurà de controlar en zones on s'accedeix a l'entorn de dibuix i variables globlals compartides.

A més a questa fase s'ha d'afegir un contador que es vagi incrementant segons l'usuarí utilitzi una tecla de moviment i també mostrar els moviments que falten per acabar.

Per altra banda, hem de introduir un sincronisme que quan l'usuari faci ús de la tecla espai es paralitzin tot tipus de moviments i el rellotge continui contant fins que torni a premer l'espai.

També hem de implementar que quan s'indiqui 0 moviments per l'usuari el programa haurá donará a l'usuari un numero de moviments infinit.

2.2. Disseny

Pel disseny d'aquesta fase amb l'ús de semàfors per sincronitzar els threads primerament hem creat 2 mutex per gestionar zones crítiques.

Hem afegit una nova implementació que servirà per pausar el joc quan es premi la tecla espai.

Hem implementat a tot el codi on hi han zones crítquites aquestes dues funcions Pthread_mutex_lock(&mutex) i Pthread_mutex_unlock(&mutex), per poder fer semàfors.

A la funció rellotge ara controlem per a que mostri per pantalla els moviments realitzats i els restants fent una resta dels inicials menys els moviments que porta. A més gestionem que si s'indica que es vol 0 movimets, posem la variable moviments a un numero mol gran per que mai acabi.

2.3. Implementació

Tennis2.c:

```
#include <stdio.h>                      /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include "winsuport.h"                  /* incloure definicions de funcions propies */
#include <time.h>
#include <pthread.h>
#include <stdint.h> /* definició de intptr_t per màquines de 64 bits */
#include <unistd.h>

#define MIN_FIL 7                       /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80
#define MIN_PAL 3
#define MIN_VEL -1.0
#define MAX_VEL 1.0
#define MIN_RET 0.0
#define MAX_RET 5.0
#define MAX_PALETES 9
#define MAX_THREADS 12

/* variables globals */
pthread_t tid[MAX_THREADS];
float po_pf[MAX_PALETES];               /* pos. vertical de la paleta de l'ordinador, en valor real */
int ipopf[MAX_PALETES], ipopc[MAX_PALETES];         /* posicio del la paleta de l'ordinador */
float vpal[MAX_PALETES];
float palret[MAX_PALETES];
int paletes;
int temps = 0;

int n_fil, n_col, m_por;        /* dimensions del taulell i porteries */
int l_pal;                      /* longitud de les paletes */
float v_pal;                    /* velocitat de la paleta del programa */
float pal_ret;                  /* percentatge de retard de la paleta */

int ipu_pf, ipu_pc;             /* posicio del la paleta d'usuari */
int ipo_pf, ipo_pc;             /* posicio del la paleta de l'ordinador */

int ipil_pf, ipil_pc;           /* posicio de la pilota, en valor enter */
float pil_pf, pil_pc;           /* posicio de la pilota, en valor real */
```

```
float pil_vf, pil_vc;                   /* velocitat de la pilota, en valor real*/
float pil_ret;                          /* percentatge de retard de la pilota */

int retard;                     /* valor del retard de moviment, en mil.lisegons */
int moviments, moviments_inicials;              /* numero max de moviments paletes per acabar el joc */

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex_temps = PTHREAD_MUTEX_INITIALIZER;
int cont, tec;
int ilimitats;



/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins un fitxer de text, el nom del qual es passa per referencia en   */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-     */
/* tinent al SO (segons comentaris del principi del programa).             */
void carrega_parametres(const char *nom_fit)
{
 FILE *fit;

 fit = fopen(nom_fit,"rt");             /* intenta obrir fitxer */
 if (fit == NULL)
 {              fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
                exit(2);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %d %d\n",&n_fil,&n_col,&m_por,&l_pal);
 if ((n_fil < MIN_FIL) || (n_fil > MAX_FIL) ||
(n_col < MIN_COL) || (n_col > MAX_COL) || (m_por < 0) ||
 (m_por > n_fil-3) || (l_pal < MIN_PAL) || (l_pal > n_fil-3))
 {
fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");
fprintf(stderr,"\t%d =< n_fil (%d) =< %d\n",MIN_FIL,n_fil,MAX_FIL);
fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);
fprintf(stderr,"\t0 =< m_por (%d) =< n_fil-3 (%d)\n",m_por,(n_fil-3));
fprintf(stderr,"\t%d =< l_pal (%d) =< n_fil-3 (%d)\n",MIN_PAL,l_pal,(n_fil-3));
fclose(fit);
exit(3);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %f %f %f\n",&ipil_pf,&ipil_pc,&pil_vf,&pil_vc,&pil_ret);
```

```
 if ((ipil_pf < 1) || (ipil_pf > n_fil-3) ||
(ipil_pc < 1) || (ipil_pc > n_col-2) ||
(pil_vf < MIN_VEL) || (pil_vf > MAX_VEL) ||
(pil_vc < MIN_VEL) || (pil_vc > MAX_VEL) ||
(pil_ret < MIN_RET) || (pil_ret > MAX_RET))
 {
fprintf(stderr,"Error: parametre pilota incorrectes:\n");
fprintf(stderr,"\t1 =< ipil_pf (%d) =< n_fil-3 (%d)\n",ipil_pf,(n_fil-3));
fprintf(stderr,"\t1 =< ipil_pc (%d) =< n_col-2 (%d)\n",ipil_pc,(n_col-2));
fprintf(stderr,"\t%.1f =< pil_vf (%.1f) =< %.1f\n",MIN_VEL,pil_vf,MAX_VEL);
fprintf(stderr,"\t%.1f =< pil_vc (%.1f) =< %.1f\n",MIN_VEL,pil_vc,MAX_VEL);
fprintf(stderr,"\t%.1f =< pil_ret (%.1f) =< %.1f\n",MIN_RET,pil_ret,MAX_RET);
fclose(fit);
exit(4);
 }

paletes = 0;
while (!feof(fit) && paletes < MAX_PALETES){

 fscanf(fit,"%d %d %f %f\n",&ipo_pf,&ipo_pc,&v_pal,&pal_ret);
 ipopf[paletes] = ipo_pf;
 ipopc[paletes] = ipo_pc;
 vpal[paletes] = v_pal;
 palret[paletes] = pal_ret;

 if ((ipopf[paletes] < 1) || (ipopf[paletes]+l_pal > n_fil-2) ||
(ipo_pc < 5) || (ipo_pc > n_col-2) ||
(v_pal < MIN_VEL) || (v_pal > MAX_VEL) ||
(pal_ret < MIN_RET) || (pal_ret > MAX_RET))
   {
fprintf(stderr,"Error: parametres paleta ordinador incorrectes:\n");
fprintf(stderr,"\t1 =< ipo_pf (%d) =< n_fil-l_pal-3 (%d)\n",ipo_pf,(n_fil-l_pal-3));
fprintf(stderr,"\t5 =< ipo_pc (%d) =< n_col-2 (%d)\n",ipo_pc,(n_col-2));
fprintf(stderr,"\t%.1f =< v_pal (%.1f) =< %.1f\n",MIN_VEL,v_pal,MAX_VEL);
fprintf(stderr,"\t%.1f =< pal_ret (%.1f) =< %.1f\n",MIN_RET,pal_ret,MAX_RET);
fclose(fit);
exit(5);
   }
   paletes++;
}
 fclose(fit);                                        /* fitxer carregat: tot OK! */
```

```
}



/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
int inicialitza_joc(void)
{
 int i, i_port, f_port, retwin;
 char strin[51];

 retwin = win_ini(&n_fil,&n_col,'+',INVERS);   /* intenta crear taulell */

 if (retwin < 0)      /* si no pot crear l'entorn de joc amb les curses */
 { fprintf(stderr,"Error en la creacio del taulell de joc:\t");
   switch (retwin)
   {  case -1: fprintf(stderr,"camp de joc ja creat!\n");
              break;
      case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
 break;
      case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
              break;
      case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
              break;
    }
   return(retwin);
 }

 i_port = n_fil/2 - m_por/2;      /* crea els forats de la porteria */
 if (n_fil%2 == 0) i_port--;
 if (i_port == 0) i_port=1;
 f_port = i_port + m_por -1;
 for (i = i_port; i <= f_port; i++)
 {                    win_escricar(i,0,' ',NO_INV);
win_escricar(i,n_col-1,' ',NO_INV);
 }

 ipu_pf = n_fil/2; ipu_pc = 3;                                        /* inicialitzar pos. paletes */
 if (ipu_pf+l_pal >= n_fil-3) ipu_pf = 1;
 for (i=0; i< l_pal; i++)               /* dibuixar paleta inicialment */
 {
   win_escricar(ipu_pf +i, ipu_pc, '0',INVERS);
 for (int x=0; x<paletes; x++){
    win_escricar(ipopf[x] +i, ipopc[x], '1',INVERS);
```

```
    po_pf[x] = ipopf[x];
   }
  }

  pil_pf = ipil_pf; pil_pc = ipil_pc;              /* fixar valor real posicio pilota */
  win_escricar(ipil_pf, ipil_pc, '.',INVERS); /* dibuix inicial pilota */
  return(0);
}


/* funcio per moure la pilota; retorna un valor amb alguna d'aquestes                */
/* possibilitats:                                                                    */
/*                  -1 ==> la pilota no ha sortit del taulell                        */
/*                   0 ==> la pilota ha sortit per la porteria esquerra              */
/*                  >0 ==> la pilota ha sortit per la porteria dreta                 */
void * moure_pilota(void * cap)
{
  int f_h, c_h;
  char rh,rv,rd,pd;

do{
  f_h = pil_pf + pil_vf;                     /* posicio hipotetica de la pilota */
  c_h = pil_pc + pil_vc;
  cont = -1;                         /* inicialment suposem que la pilota no surt */
  rh = rv = rd = pd = ' ';
  if ((f_h != ipil_pf) || (c_h != ipil_pc))
  {                                  /* si posicio hipotetica no coincideix amb la pos. actual */
    if (f_h != ipil_pf)                      /* provar rebot vertical */
    {
      pthread_mutex_lock(&mutex);
      rv = win_quincar(f_h,ipil_pc);         /* veure si hi ha algun obstacle */
    if (rv != ' ')                           /* si no hi ha res */
       {  pil_vf = -pil_vf;                  /* canvia velocitat vertical */
       f_h = pil_pf+pil_vf;        /* actualitza posicio hipotetica */
       }
      pthread_mutex_unlock(&mutex);
    }
    if (c_h != ipil_pc)                      /* provar rebot horitzontal */
    {
      pthread_mutex_lock(&mutex);
      rh = win_quincar(ipil_pf,c_h);         /* veure si hi ha algun obstacle */
       if (rh != ' ')                        /* si no hi ha res */
```

```
      {   pil_vc = -pil_vc;                    /* canvia velocitat horitzontal */
          c_h = pil_pc+pil_vc;  /* actualitza posicio hipotetica */
      }
    pthread_mutex_unlock(&mutex);
  }
  if ((f_h != ipil_pf) && (c_h != ipil_pc))       /* provar rebot diagonal */
   {              pthread_mutex_lock(&mutex); rd = win_quincar(f_h,c_h);
     if (rd != ' ')                                        /* si no hi ha obstacle */
     {   pil_vf = -pil_vf; pil_vc = -pil_vc;       /* canvia velocitats */
         f_h = pil_pf+pil_vf;
         c_h = pil_pc+pil_vc;                    /* actualitza posicio entera */
     }
     pthread_mutex_unlock(&mutex);
  }
  pthread_mutex_lock(&mutex);
  if (win_quincar(f_h,c_h) == ' ')              /* verificar posicio definitiva */
  {                                                                          /* si no hi ha obstacle */
win_escricar(ipil_pf,ipil_pc,' ',NO_INV);    /* esborra pilota */
pil_pf += pil_vf; pil_pc += pil_vc;
ipil_pf = f_h; ipil_pc = c_h;                 /* actualitza posicio actual */
if ((ipil_pc > 0) && (ipil_pc <= n_col))      /* si no surt */
win_escricar(ipil_pf,ipil_pc,'.',INVERS); /* imprimeix pilota */
else{
  if (ipil_pc > n_col) cont = 0;
  else if(ipil_pc < 0) cont = 1;
 }
  }
  pthread_mutex_unlock(&mutex);
 }
 else { pil_pf += pil_vf; pil_pc += pil_vc; }

 win_retard(retard);

}while ((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));

}

/* funcio per moure la paleta de l'usuari en funcio de la tecla premuda */
void * mou_paleta_usuari(void * cap)
{
 do{
```

```
  pthread_mutex_lock(&mutex);
  tec=win_gettec();

  if ((tec == TEC_AVALL) && (win_quincar(ipu_pf+l_pal,ipu_pc) == ' '))
  {
    win_escricar(ipu_pf,ipu_pc,' ',NO_INV);   /* esborra primer bloc */
    ipu_pf++;                                                    /* actualitza posicio */
    win_escricar(ipu_pf+l_pal-1,ipu_pc,'0',INVERS); /* impri. ultim bloc */
    if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
  }
  if ((tec == TEC_AMUNT) && (win_quincar(ipu_pf-1,ipu_pc) == ' '))
  {
    win_escricar(ipu_pf+l_pal-1,ipu_pc,' ',NO_INV); /* esborra ultim bloc */
    ipu_pf--;                                                    /* actualitza posicio */
    win_escricar(ipu_pf,ipu_pc,'0',INVERS);                 /* imprimeix primer bloc */
    if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
  }
  if (tec == TEC_ESPAI){
    do{
      tec = win_gettec();
    }while (tec != TEC_ESPAI);
  }
  pthread_mutex_unlock(&mutex);
  win_retard(retard);

  }while((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));
}



/* funcio per moure la paleta de l'ordinador autonomament, en funcio de la */
/* velocitat de la paleta (variable global v_pal) */
void * mou_paleta_ordinador(void * index)
{
  int f_h[MAX_PALETES];
/* char rh,rv,rd; */

  int i = (intptr_t) index;
  do{
     pthread_mutex_lock(&mutex);
     f_h[i] = po_pf[i] + vpal[i];                    /* posicio hipotetica de la paleta */
     pthread_mutex_unlock(&mutex);
     if (f_h[i] != ipopf[i])          /* si pos. hipotetica no coincideix amb pos. actual */
```

```
    {
     pthread_mutex_lock(&mutex);
     if (vpal[i] > 0.0)                                    /* verificar moviment cap avall */
     {
    if (win_quincar(f_h[i]+l_pal-1,ipopc[i]) == ' ')   /* si no hi ha obstacle */
    {
                        win_escricar(ipopf[i],ipopc[i],' ',NO_INV);     /* esborra primer bloc */
     po_pf[i] += vpal[i]; ipopf[i] = po_pf[i];                  /* actualitza posicio */
     win_escricar(ipopf[i]+l_pal-1,ipopc[i],'1',INVERS); /* impr. ultim bloc */
     }
    else                                /* si hi ha obstacle, canvia el sentit del moviment */
     vpal[i] = -vpal[i];
     }
     else                                          /* verificar moviment cap amunt */
     {
     if (win_quincar(f_h[i],ipopc[i]) == ' ')      /* si no hi ha obstacle */
     {
       win_escricar(ipopf[i]+l_pal-1,ipopc[i],' ',NO_INV); /* esbo. ultim bloc */
       po_pf[i] += vpal[i]; ipopf[i] = po_pf[i];                /* actualitza posicio */
       win_escricar(ipopf[i],ipopc[i],'1',INVERS);              /* impr. primer bloc */
       }
    else                                /* si hi ha obstacle, canvia el sentit del moviment */
     vpal[i] = -vpal[i];
     }
     pthread_mutex_unlock(&mutex);
     }
     else{
       pthread_mutex_lock(&mutex);
       po_pf[i] += vpal[i];         /* actualitza posicio vertical real de la paleta */
       pthread_mutex_unlock(&mutex);
     }
     win_retard(retard);

  }while((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));

}

void * rellotge(void * cap) {
    cont = -1;
    char missatge[100];
    while (1) {
       pthread_mutex_lock(&mutex_temps);
```

```
      temps++;
      sleep(1);
      int segs = temps%60;
      int min = temps/60;

      if (ilimitats == 0){
       sprintf(missatge, "Movs: ilimitats|Temps: %d:%d", min, segs);
      win_escristr(missatge);
      }else{
       sprintf(missatge, "Movs: %d restants|%d realitzats|Temps: %d:%d",
            moviments, moviments_inicials-moviments, min, segs);
       win_escristr(missatge);
      }

      if (moviments == 0 || cont != -1) {
        win_fi();
        if (cont == 1)
          printf("Ha guanyat l'ordinador!\n");
        else if (cont == 0)
          printf("Ha guanyat l'usuari!\n");
        pthread_mutex_unlock(&mutex_temps);
        exit(0);
      }
      pthread_mutex_unlock(&mutex_temps);
      win_retard(retard);
    }
    return NULL;
}

/* programa principal*/
int main(int n_args, const char *ll_args[])
{
  if ((n_args != 3) && (n_args !=4))
  {                   fprintf(stderr,"Comanda: tennis0 fit_param moviments [retard]\n");
                      exit(1);
  }
  carrega_parametres(ll_args[1]);
  moviments=atoi(ll_args[2]);
  ilimitats = moviments;
  if (moviments == 0) moviments = 2147483647; //Valor maxim de int
  moviments_inicials=moviments;
```

```
 if (n_args == 4) retard=atoi(ll_args[3]);
 else {
  retard=100;
 }

 if (inicialitza_joc() !=0)   // intenta crear el taulell de joc
  exit(4);                // aborta si hi ha algun problema amb taulell

 pthread_mutex_init(&mutex, NULL);    //Es crea el semafor

 for (int i=0; i<paletes; i++){
  pthread_create(&tid[i], NULL, mou_paleta_ordinador, (void *)(intptr_t)i);
 }
 pthread_create(&tid[paletes], NULL, mou_paleta_usuari, NULL);
 pthread_create(&tid[paletes+1], NULL, moure_pilota, NULL);
 pthread_create(&tid[paletes+2], NULL, rellotge, NULL);

 for (int i=0; i<paletes+3; i++){
  pthread_join(tid[i], NULL);
 }

 win_fi();

 pthread_mutex_destroy(&mutex);   //Es destrueix el semafor
 pthread_mutex_destroy(&mutex_temps);   //Es destrueix el semafor
 return(0);
}
```

2.4. Joc de proves

De la mateixa manera que a la fase 1 hem comprovat que els fitxers camp1.txt, camp2.txt, camp3.txt i camp4.txt continuin funcionant correctament i així a sigut, cada camp genera les paletes que s'especifiquen a cada fitxer.

En aquesta fase 2 hem hagut d'implementar diferents accions, com que quan l'usuari prem la tecla espai el joc queda pausat i el temps que comptatge se la partida segueix funcionant, i si l'usuari torna a premir la tecla espai torna a continuar el joc amb normalitat.

Altra cosa que habiem d'implementar eren els moviments infinits, i quan s'especifica 0 moviments el joc dona moviments infinits a l'usuari correctament i només acabarà si algú dels dos fa gol o si l'usuari prem la tecla RETURN.

Per últim hem comprovat que per pantalla es mostri els moviments que porta l'usuari i els restants per acabar el joc.

3. Fase 3

3.1. Especificacions

L'objectiu principal d'aquesta fase es controlar el funcionament dels multiples processos, per tant s'ens proprosa que aprofitem la fase anterior i on habien threads eliminar-los y utilitzar processos .

Com cada paleta del ordinador será gestionada per un procés fill, aquest procés fill de la funció mou_paleta_ordinador haurá d'estar en un fitxer diferent al fitxer excecutable del main que sería el procés pare.

Per garantir el funcionament de la fase 3 haurem de controlar de manera correcta la creacio de processos més kes zonez on fem ús de la memoria compartida.

En aquesta part de la practica per poder accedir a pantalla tindrem un grup de rutines implementades a 'winsuport2.h' amb les que podem fer que la pantalla es pugui utilitzar des de els diferents processos.

Haurem de reservar memoria per emmagatzemar el contigunt del camp de joc , crear una zona de memoria compartida per la finestra i altre que ens permeti el accés al procés pare per poder inicialitzar el contingut del que es mostra per pantalla . També tindrem dues funcions que actualitzaran el contingut de la finestra i altre que eliminarà la zona de memoria compartida.

3.2. Disseny

Primer de tot per a la fase 3 hem hagut de eliminar tots els semafors, on hi habien bloqueigs de mutex els hem tret i així poder implementar processos.

Hem creat variables per poder controlar la memoria compartida, tota aquella inforamació que haurem de utilitzar en el moure_paleta_ordinador, l'hem de crear i mapejar la memoria compartida.

Després hem cambiat el tipus de les variables que eres int o float a char i d'aquesta manera poder utilitzar-los al execlp.

A continuació es crea el procés fill amb el fork(), on al execlp() s'executa el fitxer pal_ord3 on estaría la funcio mou_paleta_ordinador, i le pasem els parametres que hem compartit en memoria.

I simplement al fitxer pal_ord3.c hem copiat la funció mou_paleta_ordinador i l'hem implementat al principi identificadors de memoria compartida i altres que seran punters a les regions de la memoria compartida corresponent.

Es llegeixen els arguments pasats pel execpl() del tennis3.c i es mapeja cada identificador a cada regió de memoria compartida.

3.3. Implementació

Tennis3.c:

```
#include <stdio.h>                      /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include "winsuport2.h"                 /* incloure definicions de funcions propies */
#include <time.h>
#include <pthread.h>
#include <stdint.h> /* definició de intptr_t per màquines de 64 bits */
#include <unistd.h>
#include <stdbool.h>
#include <sys/wait.h>
#include "memoria.h"

#define MIN_FIL 7                       /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80
#define MIN_PAL 3
#define MIN_VEL -1.0
#define MAX_VEL 1.0
#define MIN_RET 0.0
#define MAX_RET 5.0
```

```
#define MAX_PALETES 9
#define MAX_THREADS 12

/* variables globals */
pthread_t tid[MAX_THREADS];
float po_pf[MAX_PALETES];                   /* pos. vertical de la paleta de l'ordinador, en valor real */
int ipopf[MAX_PALETES], ipopc[MAX_PALETES];          /* posicio del la paleta de l'ordinador */
float vpal[MAX_PALETES];
float palret[MAX_PALETES];
int paletes;
int temps = 0;
bool pausa = false;

int n_fil, n_col, m_por;        /* dimensions del taulell i porteries */
int l_pal;                                  /* longitud de les paletes */
float v_pal;                                /* velocitat de la paleta del programa */
float pal_ret;                              /* percentatge de retard de la paleta */

int ipu_pf, ipu_pc;             /* posicio del la paleta d'usuari */
int ipo_pf, ipo_pc;             /* posicio del la paleta de l'ordinador */

int ipil_pf, ipil_pc;                       /* posicio de la pilota, en valor enter */
float pil_pf, pil_pc;                       /* posicio de la pilota, en valor real */
float pil_vf, pil_vc;                       /* velocitat de la pilota, en valor real*/
float pil_ret;                              /* percentatge de retard de la pilota */

int retard;                     /* valor del retard de moviment, en mil.lisegons */
int moviments, moviments_inicials;                  /* numero max de moviments paletes per acabar el joc */

pthread_mutex_t stop = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t pantalla = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t variables = PTHREAD_MUTEX_INITIALIZER;
int cont, tec;
int segs, min;
int id_fin, *p_fin;

pid_t tpid[MAX_PALETES];                    /* taula d'identificadors dels processos fill */

/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins un fitxer de text, el nom del qual es passa per referencia en   */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-     */
```

```
/* tinent al SO (segons comentaris del principi del programa).                    */

void carrega_parametres(const char *nom_fit)

{

 FILE *fit;


 fit = fopen(nom_fit,"rt");                    /* intenta obrir fitxer */

 if (fit == NULL)

 {                  fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);

                    exit(2);

 }


 if (!feof(fit)) fscanf(fit,"%d %d %d %d\n",&n_fil,&n_col,&m_por,&l_pal);

 if ((n_fil < MIN_FIL) || (n_fil > MAX_FIL) ||

(n_col < MIN_COL) || (n_col > MAX_COL) || (m_por < 0) ||

 (m_por > n_fil-3) || (l_pal < MIN_PAL) || (l_pal > n_fil-3))

 {

fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");

fprintf(stderr,"\t%d =< n_fil (%d) =< %d\n",MIN_FIL,n_fil,MAX_FIL);

fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);

fprintf(stderr,"\t0 =< m_por (%d) =< n_fil-3 (%d)\n",m_por,(n_fil-3));

fprintf(stderr,"\t%d =< l_pal (%d) =< n_fil-3 (%d)\n",MIN_PAL,l_pal,(n_fil-3));

fclose(fit);

exit(3);

 }


 if (!feof(fit)) fscanf(fit,"%d %d %f %f %f\n",&ipil_pf,&ipil_pc,&pil_vf,&pil_vc,&pil_ret);

 if ((ipil_pf < 1) || (ipil_pf > n_fil-3) ||

(ipil_pc < 1) || (ipil_pc > n_col-2) ||

(pil_vf < MIN_VEL) || (pil_vf > MAX_VEL) ||

(pil_vc < MIN_VEL) || (pil_vc > MAX_VEL) ||

(pil_ret < MIN_RET) || (pil_ret > MAX_RET))

 {

fprintf(stderr,"Error: parametre pilota incorrectes:\n");

fprintf(stderr,"\t1 =< ipil_pf (%d) =< n_fil-3 (%d)\n",ipil_pf,(n_fil-3));

fprintf(stderr,"\t1 =< ipil_pc (%d) =< n_col-2 (%d)\n",ipil_pc,(n_col-2));

fprintf(stderr,"\t%.1f =< pil_vf (%.1f) =< %.1f\n",MIN_VEL,pil_vf,MAX_VEL);

fprintf(stderr,"\t%.1f =< pil_vc (%.1f) =< %.1f\n",MIN_VEL,pil_vc,MAX_VEL);

fprintf(stderr,"\t%.1f =< pil_ret (%.1f) =< %.1f\n",MIN_RET,pil_ret,MAX_RET);

fclose(fit);

exit(4);

 }
```

```
paletes = 0;
while (!feof(fit) && paletes < MAX_PALETES){

 fscanf(fit,"%d %d %f %f\n",&ipo_pf,&ipo_pc,&v_pal,&pal_ret);
 ipopf[paletes] = ipo_pf;
 ipopc[paletes] = ipo_pc;
 vpal[paletes] = v_pal;
 palret[paletes] = pal_ret;

 if ((ipopf[paletes] < 1) || (ipopf[paletes]+l_pal > n_fil-2) ||
(ipo_pc < 5) || (ipo_pc > n_col-2) ||
(v_pal < MIN_VEL) || (v_pal > MAX_VEL) ||
(pal_ret < MIN_RET) || (pal_ret > MAX_RET))
   {
fprintf(stderr,"Error: parametres paleta ordinador incorrectes:\n");
fprintf(stderr,"\t1 =< ipo_pf (%d) =< n_fil-l_pal-3 (%d)\n",ipo_pf,(n_fil-l_pal-3));
fprintf(stderr,"\t5 =< ipo_pc (%d) =< n_col-2 (%d)\n",ipo_pc,(n_col-2));
fprintf(stderr,"\t%.1f =< v_pal (%.1f) =< %.1f\n",MIN_VEL,v_pal,MAX_VEL);
fprintf(stderr,"\t%.1f =< pal_ret (%.1f) =< %.1f\n",MIN_RET,pal_ret,MAX_RET);
fclose(fit);
exit(5);
   }
   paletes++;
}
 fclose(fit);                                    /* fitxer carregat: tot OK! */

}


/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
int inicialitza_joc(void)
{
 int i, i_port, f_port, retwin;
 char strin[51];
 cont = -1;
 retwin = win_ini(&n_fil,&n_col,'+',INVERS);   /* intenta crear taulell */

 id_fin = ini_mem(retwin);
 p_fin = map_mem(id_fin);

 win_set(p_fin, n_fil, n_col);
```

```
 if (retwin < 0)      /* si no pot crear l'entorn de joc amb les curses */
 { fprintf(stderr,"Error en la creacio del taulell de joc:\t");
   switch (retwin)
   {  case -1: fprintf(stderr,"camp de joc ja creat!\n");
              break;
      case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
break;
      case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
              break;
      case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
              break;
    }
    return(retwin);
 }


 i_port = n_fil/2 - m_por/2;      /* crea els forats de la porteria */
 if (n_fil%2 == 0) i_port--;
 if (i_port == 0) i_port=1;
 f_port = i_port + m_por -1;
 for (i = i_port; i <= f_port; i++)
 {                    pthread_mutex_lock(&pantalla);
   win_escricar(i,0,' ',NO_INV);
win_escricar(i,n_col-1,' ',NO_INV);
 pthread_mutex_unlock(&pantalla);
 }


 ipu_pf = n_fil/2; ipu_pc = 3;                                        /* inicialitzar pos. paletes */
 if (ipu_pf+l_pal >= n_fil-3) ipu_pf = 1;
 for (i=0; i< l_pal; i++)              /* dibuixar paleta inicialment */
 {                    pthread_mutex_lock(&pantalla);
   win_escricar(ipu_pf +i, ipu_pc, '0',INVERS);
   pthread_mutex_unlock(&pantalla);
 for (int x=0; x<paletes; x++){
     pthread_mutex_lock(&pantalla);
     win_escricar(ipopf[x] +i, ipopc[x], '1',INVERS);
     pthread_mutex_unlock(&pantalla);
     pthread_mutex_lock(&variables);
     po_pf[x] = ipopf[x];
     pthread_mutex_unlock(&variables);
   }
 }
```

```
  pthread_mutex_lock(&variables);
  pil_pf = ipil_pf; pil_pc = ipil_pc;          /* fixar valor real posicio pilota */
  pthread_mutex_unlock(&variables);
  pthread_mutex_lock(&pantalla);
  win_escricar(ipil_pf, ipil_pc, '.',INVERS); /* dibuix inicial pilota */
  pthread_mutex_unlock(&pantalla);
  return(0);
}


/* funcio per moure la pilota; retorna un valor amb alguna d'aquestes          */
/* possibilitats:                                                              */
/*          -1 ==> la pilota no ha sortit del taulell                          */
/*           0 ==> la pilota ha sortit per la porteria esquerra                */
/*          >0 ==> la pilota ha sortit per la porteria dreta                   */
void * moure_pilota(void * cap)
{
  int f_h, c_h;
  char rh,rv,rd,pd;
  win_set(p_fin, n_fil, n_col);
do{
  pthread_mutex_lock(&stop);
  pthread_mutex_unlock(&stop);
  f_h = pil_pf + pil_vf;                       /* posicio hipotetica de la pilota */
  c_h = pil_pc + pil_vc;
  cont = -1;                        /* inicialment suposem que la pilota no surt */
  rh = rv = rd = pd = ' ';
  if ((f_h != ipil_pf) || (c_h != ipil_pc))
  {                                 /* si posicio hipotetica no coincideix amb la pos. actual */
    if (f_h != ipil_pf)                        /* provar rebot vertical */
    {
      pthread_mutex_lock(&pantalla);
      rv = win_quincar(f_h,ipil_pc);           /* veure si hi ha algun obstacle */
      pthread_mutex_unlock(&pantalla);
    if (rv != ' ')                             /* si no hi ha res */
       {  pil_vf = -pil_vf;                    /* canvia velocitat vertical */
        f_h = pil_pf+pil_vf;         /* actualitza posicio hipotetica */
        }
    }
    if (c_h != ipil_pc)                        /* provar rebot horitzontal */
    {
      pthread_mutex_lock(&pantalla);
```

```
    rh = win_quincar(ipil_pf,c_h);              /* veure si hi ha algun obstacle */
    pthread_mutex_unlock(&pantalla);
     if (rh != ' ')                             /* si no hi ha res */
      {   pil_vc = -pil_vc;                     /* canvia velocitat horitzontal */
          c_h = pil_pc+pil_vc;  /* actualitza posicio hipotetica */
      }
  }
  if ((f_h != ipil_pf) && (c_h != ipil_pc))     /* provar rebot diagonal */
   {              pthread_mutex_lock(&pantalla); rd = win_quincar(f_h,c_h);
    pthread_mutex_unlock(&pantalla);
     if (rd != ' ')                                         /* si no hi ha obstacle */
     {   pil_vf = -pil_vf; pil_vc = -pil_vc;    /* canvia velocitats */
         f_h = pil_pf+pil_vf;
         c_h = pil_pc+pil_vc;                   /* actualitza posicio entera */
     }
  }
   pthread_mutex_lock(&pantalla);
   if (win_quincar(f_h,c_h) == ' ')             /* verificar posicio definitiva */
   {                                                                    /* si no hi ha obstacle */
win_escricar(ipil_pf,ipil_pc,' ',NO_INV);   /* esborra pilota */
pil_pf += pil_vf; pil_pc += pil_vc;
ipil_pf = f_h; ipil_pc = c_h;               /* actualitza posicio actual */
if ((ipil_pc > 0) && (ipil_pc <= n_col))    /* si no surt */
win_escricar(ipil_pf,ipil_pc,'.',INVERS); /* imprimeix pilota */
else{
   if (ipil_pc > n_col) cont = 0;
   else if(ipil_pc < 0) cont = 1;
 }
   }
   pthread_mutex_unlock(&pantalla);
 }
 else { pil_pf += pil_vf; pil_pc += pil_vc; }
 win_update();
 win_retard(retard);

}while ((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));
return NULL;
}



/* funcio per moure la paleta de l'usuari en funcio de la tecla premuda */
void * mou_paleta_usuari(void * cap)
```

```
{
  win_set(p_fin, n_fil, n_col);
  do{
  pthread_mutex_lock(&variables);
  tec=win_gettec();
  pthread_mutex_unlock(&variables);
  if ((tec == TEC_AVALL) && (win_quincar(ipu_pf+l_pal,ipu_pc) == ' ') && !pausa)
  {
    pthread_mutex_lock(&pantalla);
    win_escricar(ipu_pf,ipu_pc,' ',NO_INV);   /* esborra primer bloc */
    ipu_pf++;                                                    /* actualitza posicio */
    win_escricar(ipu_pf+l_pal-1,ipu_pc,'0',INVERS); /* impri. ultim bloc */
    pthread_mutex_unlock(&pantalla);
    if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
  }
  if ((tec == TEC_AMUNT) && (win_quincar(ipu_pf-1,ipu_pc) == ' ') && !pausa)
  {
    pthread_mutex_lock(&pantalla);
    win_escricar(ipu_pf+l_pal-1,ipu_pc,' ',NO_INV); /* esborra ultim bloc */
    ipu_pf--;                                                    /* actualitza posicio */
    win_escricar(ipu_pf,ipu_pc,'0',INVERS);                  /* imprimeix primer bloc */
    pthread_mutex_unlock(&pantalla);
    if (moviments > 0) moviments--;    /* he fet un moviment de la paleta */
  }
  if (tec == TEC_ESPAI){
    //do{
    //  tec = win_gettec();
    //}while (tec != TEC_ESPAI);
    if (!pausa) pthread_mutex_lock(&stop);
    else pthread_mutex_unlock(&stop);
    pausa = !pausa;
  }
  if (tec == TEC_RETURN){
    win_fi();
    printf("Joc terminat correctament.\n");
    printf("Has jugat %d:%d minuts\n", min, segs);
    exit(0);
  }
  win_update();
  win_retard(retard);

  }while((tec != TEC_RETURN) && (cont==-1) && ((moviments > 0) || moviments == -1));
```

```
  return NULL;
}

void * rellotge(void * cap) {
   cont = -1;
   char missatge[100];
   win_set(p_fin, n_fil, n_col);
   while (1) {
      temps++;
      sleep(1);
      segs = temps%60;
      min = temps/60;

      if (moviments == -1){
       sprintf(missatge, "Movs: ilimitats|Temps: %d:%d", min, segs);
      pthread_mutex_lock(&pantalla);
      win_escristr(missatge);
      pthread_mutex_unlock(&pantalla);
      }else{
       sprintf(missatge, "Movs: %d restants|%d realitzats|Temps: %d:%d",
            moviments, moviments_inicials-moviments, min, segs);
       pthread_mutex_lock(&pantalla);
       win_escristr(missatge);
       pthread_mutex_unlock(&pantalla);
      }

      if (moviments == 0 || cont != -1) {
         win_fi();
         if (cont == 1)
            printf("Ha guanyat l'ordinador!\n");
         else if (cont == 0)
            printf("Ha guanyat l'usuari!\n");
         printf("Has jugat %d:%d minuts\n", min, segs);
         exit(0);
      }
      win_update();
      win_retard(retard);
   }
   return NULL;
}

/* programa principal*/
```

```
int main(int n_args, const char *ll_args[])

{

  int i;

  int id_tec, id_ret, id_cont, id_moviments;          //Identificadors de variables globals

  int *p_tec, *p_ret, *p_cont, *p_moviments; //Punters de variables globals

  int id_lpal, id_ipopc, id_ipopf, id_popf, id_vpal, id_palret;  //Identificadors de paràmetres de paletes

  int *p_ipopf, *p_ipopc, *p_lpal;                  //Punters a paràmetres enters de paletes

  float *p_palret, *p_vpal, *p_popf;                //Punters a paràmetres decimals de paletes

  char tecParam[50], retParam[50], contParam[50], movParam[50], lpalParam[50], ipopcParam[50], ipopfParam[50], popfParam[50], vpalParam[50], n_filParam[50],
n_colParam[50], finParam[50], palretParam[50], index[50];


  if ((n_args != 3) && (n_args !=4))

  {                   fprintf(stderr,"Comanda: tennis0 fit_param moviments [retard]\n");

                      exit(1);

  }

  carrega_parametres(ll_args[1]);

  moviments=atoi(ll_args[2]);

  if (moviments == 0) moviments = -1;

  moviments_inicials = moviments;


  if (n_args == 4) retard=atoi(ll_args[3]);

  else {

    retard=100;

  }


  if (inicialitza_joc() !=0)   // intenta crear el taulell de joc

    exit(4);                  // aborta si hi ha algun problema amb taulell


  pthread_mutex_init(&variables, NULL);    //Es crea el semafor

  pthread_mutex_init(&pantalla, NULL);    //Es crea el semafor

  pthread_mutex_init(&stop, NULL);    //Es crea el semafor


  win_update();


  id_cont = ini_mem(sizeof(int));

  p_cont = map_mem(id_cont);

  *p_cont = cont;


  id_tec = ini_mem(sizeof(int));

  p_tec = map_mem(id_tec);

  *p_tec = tec;


  id_moviments = ini_mem(sizeof(int));
```

```
p_moviments = map_mem(id_moviments);
*p_moviments = moviments;

id_ret = ini_mem(sizeof(int));
p_ret = map_mem(id_ret);
*p_ret = retard;

id_lpal = ini_mem(sizeof(int));
p_lpal = map_mem(id_lpal);
*p_lpal = l_pal;

id_ipopf = ini_mem(sizeof(int) * paletes);
p_ipopf = map_mem(id_ipopf);
for(i = 0; i < paletes; i++) p_ipopf[i] = ipopf[i];

id_ipopc = ini_mem(sizeof(int) * paletes);
p_ipopc = map_mem(id_ipopc);
for(i = 0; i < paletes; i++) p_ipopc[i] = ipopc[i];

id_popf = ini_mem(sizeof(float) * paletes);
p_popf = map_mem(id_popf);
for (i = 0; i < paletes; i++) p_popf[i] = po_pf[i];

id_vpal = ini_mem(sizeof(float) * paletes);
p_vpal = map_mem(id_vpal);
for (i = 0; i < paletes; i++) p_vpal[i] = vpal[i];

id_palret = ini_mem(sizeof(float) * paletes);
p_palret = map_mem(id_palret);
for (i = 0; i < paletes; i++) p_palret[i] = palret[i];

sprintf(tecParam,"%i",id_tec);   /* convertir en string */
sprintf(retParam,"%i",id_ret);
sprintf(contParam,"%i",id_cont);
sprintf(movParam,"%i",id_moviments);
sprintf(lpalParam,"%i",id_lpal);
sprintf(ipopcParam,"%i",id_ipopc);
sprintf(ipopfParam,"%i",id_ipopf);
sprintf(popfParam,"%i",id_popf);
sprintf(vpalParam,"%i",id_vpal);
sprintf(palretParam,"%i",id_palret);
sprintf(n_filParam,"%d",n_fil);
```

```
  sprintf(n_colParam,"%d",n_col);

  sprintf(finParam,"%i",id_fin);

  int n = 0;


  for (int i=0; i<paletes; i++){

   tpid[n] = fork();                    /* crea un nou proces */

   if (tpid[n] == (pid_t) 0)            /* branca del fill */

   {

      sprintf(index,"%i",i);

      execlp("./pal_ord3", "pal_ord3", tecParam, retParam, contParam, movParam, lpalParam, ipopcParam, ipopfParam, popfParam, vpalParam, index, finParam, n_filParam,
n_colParam, palretParam, (char *)0);

      fprintf(stderr,"error: no puc executar el process fill \'pal_ord3\'\n");

      exit(0);

   }

   else if (tpid[n] > 0) n++;           /* branca del pare */

   //pthread_create(&tid[i], NULL, mou_paleta_ordinador, (void *)(intptr_t)i);

  }

  pthread_create(&tid[paletes], NULL, mou_paleta_usuari, NULL);

  pthread_create(&tid[paletes+1], NULL, moure_pilota, NULL);

  pthread_create(&tid[paletes+2], NULL, rellotge, NULL);


  for (int i=0; i<paletes+3; i++){

   if (i >= paletes) pthread_join(tid[i], NULL);

   else waitpid(tpid[i], NULL, 0);win_update();        /* espera finalitzacio d'un fill */

  }


  win_fi();


  pthread_mutex_destroy(&variables);    //Es destrueix el semafor

  pthread_mutex_destroy(&pantalla);    //Es destrueix el semafor

  pthread_mutex_destroy(&stop);


  elim_mem(id_cont);

  elim_mem(id_fin);

  elim_mem(id_ipopc);

  elim_mem(id_ipopf);

  elim_mem(id_lpal);

  elim_mem(id_moviments);

  elim_mem(id_palret);

  elim_mem(id_popf);

  elim_mem(id_ret);

  elim_mem(id_tec);

  elim_mem(id_vpal);
```

```
 return(0);
}
```

Pal_ord3.c:

```
#include <stdio.h>                          /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include "winsuport2.h"                     /* incloure definicions de funcions propies */
#include <time.h>
#include <stdint.h> /* definició de intptr_t per màquines de 64 bits */
#include <unistd.h>
#include "memoria.h"

#define MAX_PALETES 9

int main (int n_args, char *ll_args[]){

 int f_h;

 int *ipopf, *ipopc;
 float *vpal, *p_palret, *po_pf;
 int id_ipopf, id_ipopc, id_vpal, id_popf, id_palret;
 int id_cont, id_moviments, id_tec, id_lpal, id_ret, id_fin;
 int *cont, *moviments, *tec, *l_pal, *retard, *p_fin, *palret;
 int index;

/* char rh,rv,rd; */

 if (n_args < 15)
 {  fprintf(stderr,"proces: moure_paletes_ordinador  tecla  cont  moviments valorsDePaletes(5)\n");
exit(0);
 }
 id_tec = atoi(ll_args[1]);
 tec = map_mem(id_tec);

 id_ret = atoi(ll_args[2]);
 retard = map_mem(id_ret);

 id_cont = atoi(ll_args[3]);
 cont = map_mem(id_cont);

 id_moviments = atoi(ll_args[4]);
 moviments = map_mem(id_moviments);
```

```
id_lpal = atoi(ll_args[5]), id_ipopc = atoi(ll_args[6]), id_ipopf = atoi(ll_args[7]);

l_pal = map_mem(id_lpal), ipopc = map_mem(id_ipopc), ipopf = map_mem(id_ipopf);


id_vpal = atof(ll_args[9]), id_popf = atof(ll_args[8]);

vpal = map_mem(id_vpal), po_pf = map_mem(id_popf);


index = atoi(ll_args[10]);


id_fin = atoi(ll_args[11]);

p_fin = map_mem(id_fin);


int n_fil = atoi(ll_args[12]), n_col = atoi(ll_args[13]);


id_palret = atof(ll_args[14]);

palret = map_mem(id_palret);


win_set(p_fin, n_fil, n_col);

do{

  f_h = po_pf[index] + vpal[index];                    /* posicio hipotetica de la paleta */


  if (f_h != ipopf[index])      /* si pos. hipotetica no coincideix amb pos. actual */

  {

  if (vpal[index] > 0.0)                                /* verificar moviment cap avall */

  {

 if (win_quincar(f_h+(*l_pal)-1,ipopc[index]) == ' ')   /* si no hi ha obstacle */

 {

                win_escricar(ipopf[index],ipopc[index],' ',NO_INV);     /* esborra primer bloc */

  po_pf[index] = po_pf[index] + vpal[index]; ipopf[index] = po_pf[index];                    /* actualitza posicio */

  win_escricar(ipopf[index]+(*l_pal)-1,ipopc[index],'1',INVERS); /* impr. ultim bloc */

  }

 else{                          /* si hi ha obstacle, canvia el sentit del moviment */

   vpal[index] = -vpal[index];

  }

  }

  else                                          /* verificar moviment cap amunt */

  {

  if (win_quincar(f_h,ipopc[index]) == ' ')      /* si no hi ha obstacle */

  {

   win_escricar(ipopf[index]+(*l_pal)-1,ipopc[index],' ',NO_INV); /* esbo. ultim bloc */

   po_pf[index] = po_pf[index] + vpal[index]; ipopf[index] = po_pf[index];                    /* actualitza posicio */

   win_escricar(ipopf[index],ipopc[index],'1',INVERS);/* impr. primer bloc */
```

```
      }
    else{                          /* si hi ha obstacle, canvia el sentit del moviment */
      vpal[index] = -vpal[index];
    }
    }
    }
    else{
      po_pf[index] = po_pf[index] + vpal[index];          /* actualitza posicio vertical real de la paleta */
    }
    win_retard(*retard);
  }while((*tec != TEC_RETURN) && (*cont==-1) && ((*moviments > 0) || *moviments == -1));
  //}while(1);
  return(0);
}
```

3.4. Joc de proves

El joc de proves de la fase 3 es verificant els mateixos casos que la fase 2, el programa s'executa i funciona en tots els casos, només que tenim un problema en el pal_ord3.c ja que les variables que controlan la condició de final de joc no s'actualitzan i per tant el joc no acaba mai.

Quan l'usauri prem el botó de pausa es pausa la paleta de l'usuari i la pilota, pero la paleta de l'ordinador no es pausa i continua en moviment sempre.