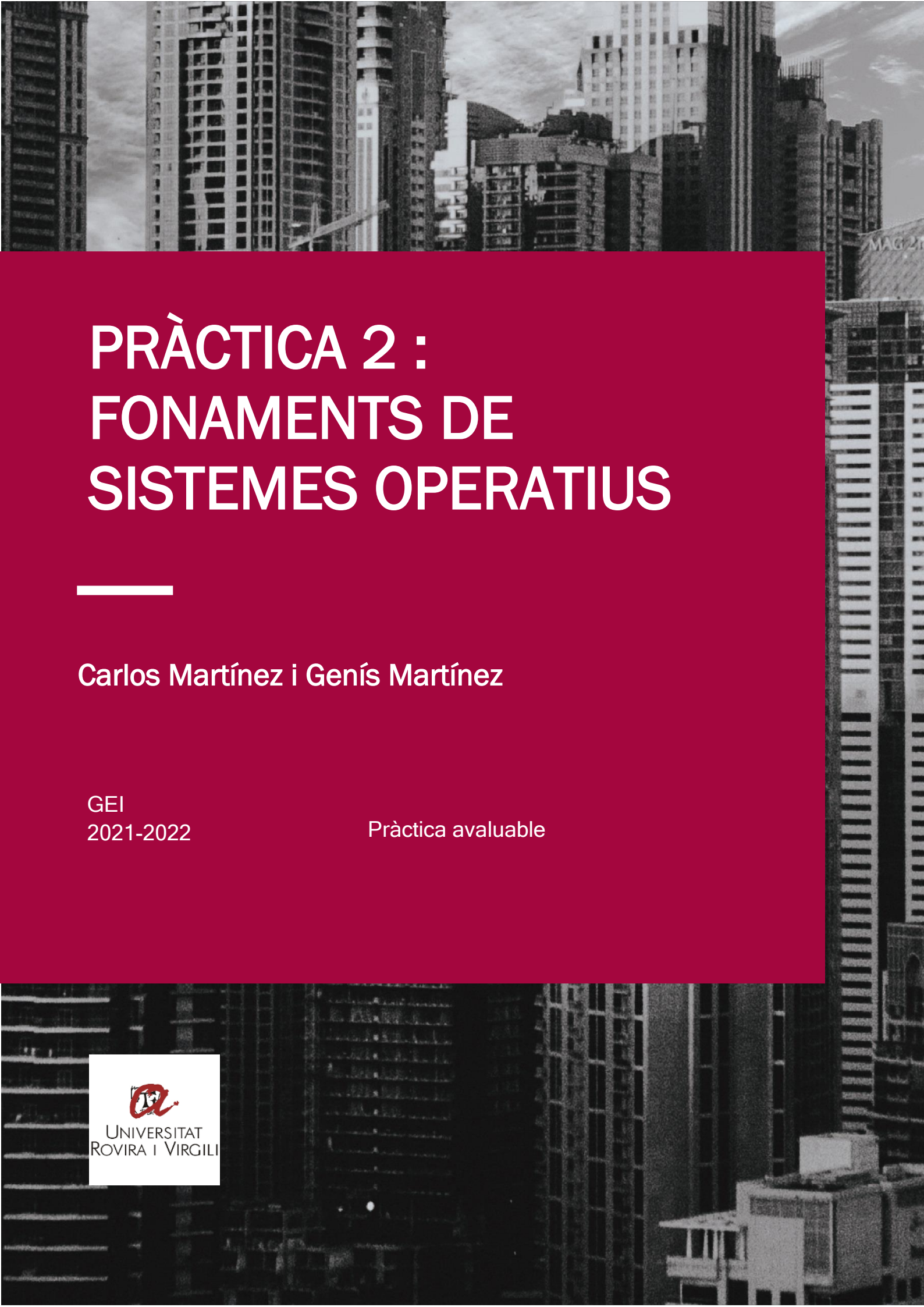


PRÀCTICA 2 : FONAMENTS DE SISTEMES OPERATIUS

Carlos Martínez i Genís Martínez

GEI
2021-2022

Pràctica avaluable

UNIVERSITAT ROVIRA I VIRGILI

Índex

1.- *Decisions de disseny*

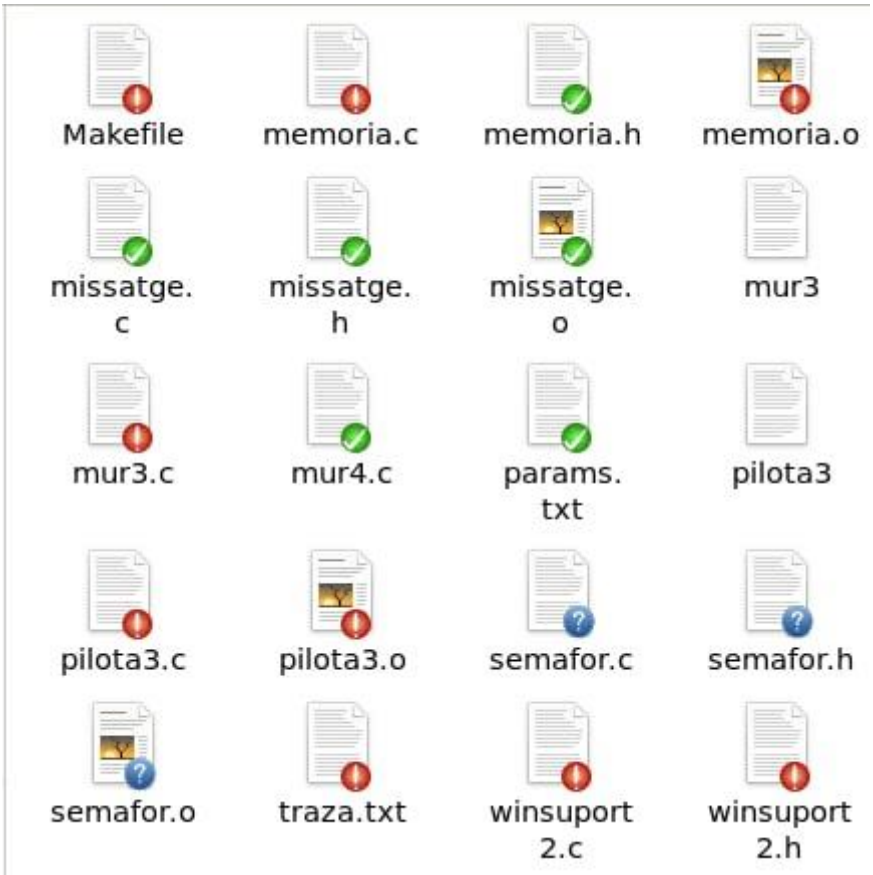
Com a decisions de disseny, hem pres moltes, però les més rellevants han sigut:

- Determinar quines variables havien de ser globals o locals
- Quan era necessari iniciar o tancar un semàfor
- Quines variables convertir a vectors
- Com iniciar la memòria compartida
- Etc.

La pràctica la hem desenvolupat mitjançant git-hub de forma remota i dividida en diverses branques depenent de les fases per poder tirar enrere en cas de possibles errors.

A l'hora d'estructurar la pràctica hem dividit el contingut en dues carpetes.

1. mur0Fase1 -> per a les fases que es realitzaven amb threads.
2. mur0Fase2 -> per a les fases que es realitzaven amb processos.

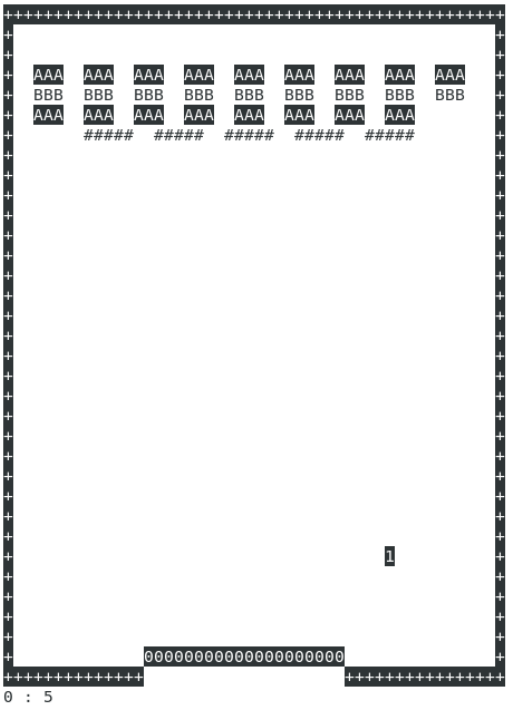


2.- *Pla de proves*

Com a joc de proves d'aquesta pràctica hem creat diferents arxius anomenats "paramsX.txt" en els quals les variables de dimensions i velocitats varien en funció del que voliem provar per a verificar el funcionament.

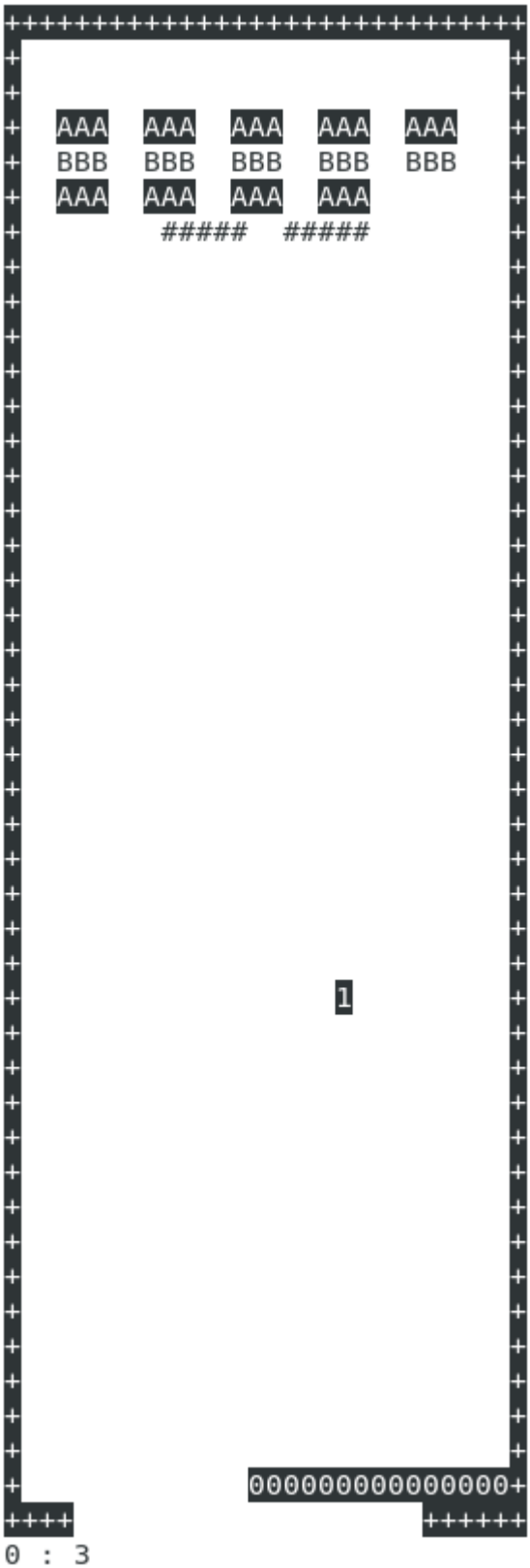
Per a realitzar aquestes proves ens hem ajudat d'un fitxer anomentat "traza.txt" en el qual anavem desglosant el codi amb diferents identificadors en les diferents parts per analitzar que succeïa en tot moment i po der detectar així possibles errors.

Prova 1:




```
35 50 20
32 14 20
21.0 20.0 -1.0 0.3
3.0 5.0 1.0 -0.5
12.0 10.0 1.0 -0.5
```

Prova 2:




```
45 30 20
42 14 15
21.0 20.0 -1.0 0.3
3.0 5.0 1.0 -0.5
12.0 10.0 1.0 -0.5
```

Prova 3:


```
0 : 2

30 70 20
26 14 15
21.0 20.0 -0.7 0.8
3.0 5.0 0.7 -0.5
12.0 10.0 1.0 -0.4
```

Makefile mur0Fase1:

mur0Fase1 > Makefile

```
mur0 : mur0.c winsuport.o winsuport.h
	gcc -Wall mur0.c winsuport.o -o mur0 -lcurses -lpthread


mur1 : mur1.c winsuport.o winsuport.h
	gcc -Wall mur1.c winsuport.o -o mur1 -lcurses -lpthread


mur2 : mur2.c winsuport.o winsuport.h
	gcc -Wall mur2.c winsuport.o -o mur2 -lcurses -lpthread


winsuport.o : winsuport.c winsuport.h
	gcc -Wall -c winsuport.c -o winsuport.o

clean :
	rm -f winsuport.o mur0 mur1 mur2
```

Makefile mur0Fase2:

mur0Fase2 > Makefile

```
memoria.o : memoria.c memoria.h
	gcc -c -Wall memoria.c -o memoria.o

winsuport2.o : winsuport2.c winsuport2.h
	gcc -Wall -c winsuport2.c -o winsuport2.o

missatge.o : missatge.c missatge.h
	gcc -c -Wall missatge.c -o missatge.o
pilota3: pilota3.c winsuport2.o memoria.o missatge.o
	gcc -Wall pilota3.c winsuport2.o memoria.o missatge.o -o pilota3 -lcurses -lpthread
mur3: mur3.c winsuport2.o memoria.o missatge.o pilota3.o
	gcc -Wall mur3.c winsuport2.o missatge.o memoria.o -o mur3 -lcurses -lpthread
mur4: mur4.c winsuport2.o memoria.o missatge.o semafor.o pilota3
	gcc -Wall mur4.c winsuport2.o memoria.o semafor.o missatge.o -g -o mur4 -lcurses -lpthread
clean :
	rm -f winsuport.o mur0 mur1 mur2 mur3 mur4 pilota3 memoria.o missatge.o
```

traza.txt mur0Fase1:

mur0Fase1 > traza.txt

```
Traza 1
Traza 2
Entro
n_fil= 30, n_col= 70, m_por= 20
Pilotes =0
Entro bucle 0
Mig
Entro bucle 1
Mig
Entro bucle 2
Mig
N_pil= 3
Surto
Traza 4
Traza 5
Traza 6
Traza Final
```

traza.txt mur0Fase2:

mur0Fase2 > traza.txt

```
Traza 1
Traza 2
Entro
Pilotes =0
Entro bucle 0
Mig
Entro bucle 1
Mig
Entro bucle 2
Mig
N_pil= 3
Surto
Traza 4
Traza 5
Inicialitza joc:
id_shm= 2555923
    final2= 2588692
    id_shm= 2588692
    fi2 mur3= 0
    id_win= 2621461
    id_shm= 2621461
id_mis= 0
id_mis= 32769
id_mis= 65538
Fi2= 0
Fi2= 0
Paleta:
Fi2= 0
Fi2= 0
```

3.- *Codi font, conclusions i autoavaluació*

Codi font:

mur1

```
/*****************************************************************************/
/*                                                                           */
/*                           mur0.c                                          */
/*                                                                           */
/*  Programa inicial d'exemple per a les practiques 2 de FSO.                */
/*                                                                           */
/*  Compilar i executar:                                                     */
/*     El programa invoca les funcions definides a "winsuport.c", les        */
/*     quals proporcionen una interficie senzilla per crear una finestra     */
/*     de text on es poden escriure caracters en posicions especifiques de   */
/*     la pantalla (basada en CURSES); per tant, el programa necessita ser   */
/*     compilat amb la llibreria 'curses':                                   */
/*                                                                           */
/*       $ gcc -Wall -c winsuport.c -o winsuport.o                           */
/*       $ gcc -Wall mur0.c winsuport.o -o mur0 -lcurses                     */
/*                                                                           */
/*  Al tenir una orientació vertical cal tenir un terminal amb prou files.   */
/*  Per exemple:                                                             */
/*               xterm -geometry 80x50                                       */
/*               gnome-terminal --geometry 80x50                             */
/*                                                                           */
/*****************************************************************************/

//#include <stdint.h>       /* intptr_t for 64bits machines */
#define _REENTRANT
#include <pthread.h>
#include <stdio.h>      /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include <string.h>
#include "winsuport.h"      /* incloure definicions de funcions propies */
#include <stdint.h>
#include <time.h>
/* definicio de constants */
#define MAX_THREADS 10
#define LENGTH 1
#define MAXBALLS 9
#define MIN_FIL 10      /* dimensions del camp. Depenen del terminal ! */
#define MAX_FIL 50
#define MIN_COL 10
#define MAX_COL 80
#define BLKSIZE 3
#define BLKGAP  2
#define BLKCHAR 'B'
```

```
#define WLLCHAR '#'
#define FRNTCHAR 'A'
#define LONGMISS    65
#define MAX_THREADS 10
pthread_t tid[MAX_THREADS];
pthread_mutex_t mutex= PTHREAD_MUTEX_INITIALIZER;
            /* variables globals */
char *descripcio[] = {
    "\n",
    "Aquest programa implementa una versio basica del joc Arkanoid o Breakout:\n",
    "generar un camp de joc rectangular amb una porteria, una paleta que s\'ha\n",
    "de moure amb el teclat per a cobrir la porteria, i una pilota que rebota\n",
    "contra les parets del camp, a la paleta i els blocs. El programa acaba si\n",
    "la pilota surt per la porteria o no queden mes blocs. Tambe es pot acabar\n",
    "amb la tecla RETURN.\n",
    "\n",
    "  Arguments del programa:\n",
    "\n",
    "       $ ./mur0 fitxer_config [retard]\n",
    "\n",
    "     El primer argument ha de ser el nom d\'un fitxer de text amb la\n",
    "     configuracio de la partida, on la primera fila inclou informacio\n",
    "     del camp de joc (valors enters), i la segona fila indica posicio\n",
    "     i velocitat de la pilota (valors reals):\n",
    "          num_files  num_columnes  mida_porteria\n",
    "          pos_fil_pal pos_col_pal mida_pal\n"
    "          pos_fila   pos_columna   vel_fila  vel_columna\n",
    "\n",
    "     on els valors minims i maxims admesos son els seguents:\n",
    "          MIN_FIL <= num_files     <= MAX_FIL\n",
    "          MIN_COL <= num_columnes  <= MAX_COL\n",
    "          0 < mida_porteria < num_files-2\n",
    "        1.0 <= pos_fila     <= num_files-3\n",
    "        1.0 <= pos_columna  <= num_columnes\n",
    "       -1.0 <= vel_fila     <= 1.0\n",
    "       -1.0 <= vel_columna  <= 1.0\n",
    "\n",
    "     Alternativament, es pot donar el valor 0 a num_files i num_columnes\n",
    "     per especificar que es vol que el tauler ocupi tota la pantalla. Si\n",
    "     tambe fixem mida_porteria a 0, el programa ajustara la mida
d\'aquesta\n",
    "     a 3/4 de l\'altura del camp de joc.\n",
    "\n",
    "     A mes, es pot afegir un segon argument opcional per indicar el\n",
    "     retard de moviment del joc en mil.lisegons; el valor minim es 10,\n",
    "     el valor maxim es 1000, i el valor per defecte d'aquest parametre\n",
    "     es 100 (1 decima de segon).\n",
    "\n",
    "  Codis de retorn:\n",
    "     El programa retorna algun dels seguents codis:\n",
    "    0  ==>  funcionament normal\n",
```

```
    "   1  ==>  numero d'arguments incorrecte\n",
    "   2  ==>  no s\'ha pogut obrir el fitxer de configuracio\n",
    "   3  ==>  algun parametre del fitxer de configuracio es erroni\n",
    "   4  ==>  no s\'ha pogut crear el camp de joc (no pot iniciar CURSES)\n",
    "\n",
    "   Per a que pugui funcionar aquest programa cal tenir instal.lada la\n",
    "   llibreria de CURSES (qualsevol versio).\n",
    "\n",
    "*"
};              /* final de la descripcio */

int n_fil, n_col;       /* numero de files i columnes del taulell */
int m_por;          /* mida de la porteria (en caracters) */
int f_pal, c_pal;       /* posicio del primer caracter de la paleta */
int m_pal;              /* mida de la paleta */
int f_pil[MAXBALLS], c_pil[MAXBALLS];       /* posicio de la pilota, en valor enter
*/
float pos_f[MAXBALLS], pos_c[MAXBALLS];     /* posicio de la pilota, en valor real
*/
float vel_f[MAXBALLS], vel_c[MAXBALLS];     /* velocitat de la pilota, en valor
real */
int nblocs = 0;
int dirPaleta = 0;
int retard;         /* valor del retard de moviment, en mil.lisegons */
int fi1,fi2;
char strin[LONGMISS];           /* variable per a generar missatges de text */
int n_pil;
int fi1=0,fi2=0;
int ind=0;
time_t temps_actual,temps_init;

/* funcio per carregar i interpretar el fitxer de configuracio de la partida */
/* el parametre ha de ser un punter a fitxer de text, posicionat al principi */
/* la funcio tanca el fitxer, i retorna diferent de zero si hi ha problemes  */
int carrega_configuracio(FILE * fit)
{
    fprintf(stderr,"Entro\n");
    int ret = 0;
    int i=0;
    fscanf(fit, "%d %d %d\n", &n_fil, &n_col, &m_por);  /* camp de joc */
    fscanf(fit, "%d %d %d\n", &f_pal, &c_pal, &m_pal);  /* paleta */
    fprintf(stderr,"Pilotes =%d\n",n_pil);
    while(!feof(fit)&&i<MAXBALLS){//||(ret<=5&&ret>=0)){
        fprintf(stderr,"Entro bucle %d\n",i);
        fscanf(fit, "%f %f %f %f\n", &pos_f[i], &pos_c[i], &vel_f[i],
&vel_c[i]);   /* pilota */

        //i++;

    if ((n_fil != 0) || (n_col != 0)) { /* si no dimensions maximes */
```

```
        if ((n_fil < MIN_FIL) || (n_fil > MAX_FIL) || (n_col < MIN_COL) || (n_col >
MAX_COL))
            ret = 1;
        else
        if (m_por > n_col - 3)
            ret = 2;
        else if ((m_pal > n_col - 3) || (m_pal < 1) || (f_pal > n_fil-1) || (f_pal
< 1) || (c_pal + m_pal > n_col -1 || c_pal < 1 )){
            ret = 3;
            fprintf(stderr,"Traza paso 3:\n");}
        else if ((pos_f[i] < 1) || (pos_f[i] >= n_fil - 3) || (pos_c[i] < 1)
             || (pos_c[i] > n_col - 1)) /* tres files especials: línia d'estat,
porteria i paleta */
            ret = 4;
    }
    fprintf(stderr,"Mig\n");
    if ((vel_f[i] < -1.0) || (vel_f[i] > 1.0) || (vel_c[i] < -1.0) || (vel_c[i] >
1.0))
        ret = 5;

    n_pil++;i++;
    }

    if (ret != 0) {     /* si ha detectat algun error */
        fprintf(stderr, "Error en fitxer de configuracio:\n");
        switch (ret) {
        case 1:
            fprintf(stderr,
                "\tdimensions del camp de joc incorrectes:\n");
            fprintf(stderr, "\tn_fil= %d \tn_col= %d\n", n_fil,
                n_col);
            break;
        case 2:
            fprintf(stderr, "\tmida de la porteria incorrecta:\n");
            fprintf(stderr, "\tm_por= %d\n", m_por);
            break;
        case 3:
            fprintf(stderr,"\tmida de la paleta incorrecta:\n");
            fprintf(stderr, "\tf_pal= %d \tc_pal= %d \t m_pal=%d\n",
f_pal,c_pal,m_pal);
            break;
        case 4:
            fprintf(stderr, "\tposicio de la pilota incorrecta:\n");
            fprintf(stderr, "\tpos_f= %.2f \tpos_c= %.2f\n", pos_f[i], pos_c[i]);
            break;
        case 5:
            fprintf(stderr,
                "\tvelocitat de la pilota incorrecta:\n");
            fprintf(stderr, "\tvel_f= %.2f \tvel_c= %.2f\n", vel_f[i],
                vel_c[i]);
            break;
```

```
        }
    }
    fclose(fit);
    fprintf(stderr,"Surto\n");
    return (ret);
}

/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
/* retorna diferent de zero si hi ha algun problema */
int inicialitza_joc(void)
{
    int i, retwin;
    int i_port, f_port; /* inici i final de porteria */
    int c, nb, offset;

    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);
    }

    if (m_por > n_col - 2)
        m_por = n_col - 2;  /* limita valor de la porteria */
    if (m_por == 0)
        m_por = 3 * (n_col - 2) / 4;    /* valor porteria per defecte */

    i_port = n_col / 2 - m_por / 2 - 1; /* crea el forat de la porteria */
    f_port = i_port + m_por - 1;
    for (i = i_port; i <= f_port; i++){
        win_escricar(n_fil - 2, i, ' ', NO_INV);

    }

    n_fil = n_fil - 1;  /* descompta la fila de missatges */
```

```
    for (i = 0; i < m_pal; i++) /* dibuixar paleta inicial */{
        win_escricar(f_pal, c_pal + i, '0', INVERS);

    }
    for(i=0;i<n_pil;i++){
    /* generar la pilota */
    if (pos_f[i] > n_fil - 1)
        pos_f[i] = n_fil - 1;   /* limita posicio inicial de la pilota */
    if (pos_c[i] > n_col - 1)
        pos_c[i] = n_col - 1;
    f_pil[i] = pos_f[i];
    c_pil[i] = pos_c[i];        /* dibuixar la pilota inicialment */
    win_escricar(f_pil[i], c_pil[i], '1'+i, INVERS);

    }
    // generar els blocs
    nb = 0;
    nblocs = n_col / (BLKSIZE + BLKGAP) - 1;
    offset = (n_col - nblocs * (BLKSIZE + BLKGAP) + BLKGAP) / 2;
    for (i = 0; i < nblocs; i++) {
        for (c = 0; c < BLKSIZE; c++) {
            win_escricar(3, offset + c, FRNTCHAR, INVERS);
            nb++;
            win_escricar(4, offset + c, BLKCHAR, NO_INV);
            nb++;
            win_escricar(5, offset + c, FRNTCHAR, INVERS);
            nb++;
        }
        offset += BLKSIZE + BLKGAP;
    }
    nblocs = nb / BLKSIZE;
    // generar les defenses
    nb = n_col / (BLKSIZE + 2 * BLKGAP) - 2;
    offset = (n_col - nb * (BLKSIZE + 2 * BLKGAP) + BLKGAP) / 2;
    for (i = 0; i < nb; i++) {
        for (c = 0; c < BLKSIZE + BLKGAP; c++) {
            win_escricar(6, offset + c, WLLCHAR, NO_INV);
        }
        offset += BLKSIZE + 2 * BLKGAP;
    }

    sprintf(strin,
        "Tecles: \'%c\'-> Esquerra, \'%c\'-> Dreta, RETURN-> sortir\n",
        TEC_ESQUER, TEC_DRETA);
    win_escristr(strin);
    return (0);
}

/* funcio que escriu un missatge a la línia d'estat i tanca les curses */
void mostra_final(char *miss)
```

```
{   int lmarge;
    char marge[LONGMISS];

    /* centrar el misssatge */
    lmarge=(n_col+strlen(miss))/2;
    sprintf(marge,"%%%ds",lmarge);

    sprintf(strin, marge,miss);
    win_escristr(strin);

    /* espera tecla per a que es pugui veure el missatge */
    getchar();
}

/* Si hi ha una col.lisió pilota-bloci esborra el bloc */
void comprovar_bloc(int f, int c)
{
    int col;
    char quin = win_quincar(f, c);
    int pil= ind;
    if (quin == BLKCHAR || quin == FRNTCHAR) {
        col = c;
        while (win_quincar(f, col) != ' ') {
            win_escricar(f, col, ' ', NO_INV);
            col++;
        }
        col = c - 1;
        while (win_quincar(f, col) != ' ') {
            win_escricar(f, col, ' ', NO_INV);
            col--;
        }

        /* generar nova pilota ? */
        if(quin==BLKCHAR&&pil<n_pil){
            fprintf(stderr,"Comprovar bloc, n_pil[1]= %d \n",n_pil);
            win_escricar(f_pil[pil],c_pil[pil],'1',INVERS);
            pthread_create(&tid[pil],NULL,mou_pilota,(void *)(intptr_t)pil);
            ind++;
        }
        nblocs--;
    }
}



float control_impacte2(int c_pil, float velc0) {
    int distApal;
    float vel_c;

    distApal = c_pil - c_pal;
    if (distApal >= 2*m_pal/3)  /* costat dreta */
        vel_c = 0.5;
```

```
    else if (distApal <= m_pal/3)   /* costat esquerra */
        vel_c = -0.5;
    else if (distApal == m_pal/2)   /* al centre */
        vel_c = 0.0;
    else /*: rebot normal */
        vel_c = velc0;
    return vel_c;
}

/* funcio per moure la pilota: retorna un 1 si la pilota surt per la porteria,*/
/* altrament retorna un 0 */
void * mou_pilota(void * index)
{

    int f_h, c_h;
    int pil= (intptr_t) index;
    char rh, rv, rd;
    int fora = 0;

    do{
    fora=0;
    f_h = pos_f[pil] + vel_f[pil];  /* posicio hipotetica de la pilota (entera) */
    c_h = pos_c[pil] + vel_c[pil];
    rh = rv = rd = ' ';
    if ((f_h != f_pil[pil]) || (c_h != c_pil[pil])) {
    /* si posicio hipotetica no coincideix amb la posicio actual */
        if (f_h != f_pil[pil]) {    /* provar rebot vertical */

            rv = win_quincar(f_h, c_pil[pil]);  /* veure si hi ha algun obstacle */
            if (rv != ' ') {    /* si hi ha alguna cosa */
                comprovar_bloc(f_h, c_pil[pil]);
                if (rv == '0')  /* col.lisió amb la paleta? */
                    //control_impacte();
                    vel_c[pil] = control_impacte2(c_pil[pil], vel_c[pil]);
                vel_f[pil] = -vel_f[pil];   /* canvia sentit velocitat vertical */
                f_h = pos_f[pil] + vel_f[pil];  /* actualitza posicio hipotetica */
            }
        }
        if (c_h != c_pil[pil]) {    /* provar rebot horitzontal */
            rh = win_quincar(f_pil[pil], c_h);  /* veure si hi ha algun obstacle */
            if (rh != ' ') {    /* si hi ha algun obstacle */
                comprovar_bloc(f_pil[pil], c_h);
                /* TODO?: tractar la col.lisio lateral amb la paleta */
                vel_c[pil] = -vel_c[pil];   /* canvia sentit vel. horitzontal */
                c_h = pos_c[pil] + vel_c[pil];  /* actualitza posicio hipotetica */
            }
        }
        if ((f_h != f_pil[pil]) && (c_h != c_pil[pil])) {   /* provar rebot
diagonal */
            rd = win_quincar(f_h, c_h);
            if (rd != ' ') {    /* si hi ha obstacle */
```

```
                comprovar_bloc(f_h, c_h);
                vel_f[pil] = -vel_f[pil];
                vel_c[pil] = -vel_c[pil];   /* canvia sentit velocitats */
                f_h = pos_f[pil] + vel_f[pil];
                c_h = pos_c[pil] + vel_c[pil];  /* actualitza posicio entera */
            }
        }
        /* mostrar la pilota a la nova posició */
        if (win_quincar(f_h, c_h) == ' ') { /* verificar posicio definitiva *//* si
no hi ha obstacle */
            //pthread_mutex_lock(&mutex);
            win_escricar(f_pil[pil], c_pil[pil], ' ', NO_INV);  /* esborra pilota
*/
            pos_f[pil] += vel_f[pil];
            pos_c[pil] += vel_c[pil];
            f_pil[pil] = f_h;
            c_pil[pil] = c_h;   /* actualitza posicio actual */
            if (f_pil[pil] != (n_fil - 1)){ /* si no surt del taulell, */
                win_escricar(f_pil[pil], c_pil[pil], '1'+pil, INVERS);  /*
imprimeix pilota */
            }else{
                fi2 = 1;fora=1;
                }
        }


    } else {    /* posicio hipotetica = a la real: moure */
        pos_f[pil] += vel_f[pil];
        pos_c[pil] += vel_c[pil];
    }
    if((nblocs==0)||(fora==1)){
        fi2=1;
    }
    win_retard(retard);
    } while (!fi1 && !fi2);
    return ((void *) index);
}

/* funcio per moure la paleta segons la tecla premuda */
/* retorna un boolea indicant si l'usuari vol acabar */

void * mou_paleta(void * nul)
{
    //int tecla, result;
    int tecla;
    fprintf(stderr,"Paleta:\n");
do{
    tecla = win_gettec();
    if (tecla != 0) {
        if ((tecla == TEC_DRETA) && ((c_pal + m_pal) < n_col - 1)) {
```

```
                win_escricar(f_pal, c_pal, ' ', NO_INV);    /* esborra primer bloc
*/
                //pthread_mutex_unlock(&mutex);
                c_pal++;    /* actualitza posicio */
                //pthread_mutex_lock(&mutex);
                win_escricar(f_pal, c_pal + m_pal - 1, '0', INVERS);    /*esc.
ultim bloc */
        }
        if ((tecla == TEC_ESQUER) && (c_pal > 1)) {
                win_escricar(f_pal, c_pal + m_pal - 1, ' ', NO_INV);    /*esborra
ultim bloc */
                c_pal--;    /* actualitza posicio */
                win_escricar(f_pal, c_pal, '0', INVERS);    /* escriure primer bloc
*/

                pthread_mutex_unlock(&mutex);
        }
        if (tecla == TEC_RETURN){
            fi1 = 1;
            }
                /* final per pulsacio RETURN */
        dirPaleta = tecla;  /* per a afectar al moviment de les pilotes */
    }
    time(&temps_actual);
    win_retard(retard);
    double diferencia=difftime(temps_actual,temps_init);
    int minute=(int)diferencia/60;
    int seconds=(int)diferencia-minute*60;
    char temps[20];
    sprintf(temps,"%d : %d",minute,seconds);
    win_escristr(temps);
    }while (!fi1 && !fi2);
    return ((void*) 0);
}

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

    time(&temps_init);
    FILE *file;
    file=fopen("traza.txt","w");
    int i;
    FILE *fit_conf;

    fprintf(stderr,"Traza 1\n");

    if ((n_args != 2) && (n_args != 3)) {   /* si numero d'arguments incorrecte */
        i = 0;
        do
            fprintf(stderr, "%s", descripcio[i++]); /* imprimeix descripcio */
```

```
        while (descripcio[i][0] != '*');    /* mentre no arribi al final */
        exit(1);
    }

    fit_conf = fopen(ll_args[1], "rt"); /* intenta obrir el fitxer */

    if (!fit_conf) {
        fprintf(stderr, "Error: no s'ha pogut obrir el fitxer \'%s\'\n",
            ll_args[1]);
        exit(2);
    }

    fprintf(stderr,"Traza 2\n");

    if (carrega_configuracio(fit_conf) != 0){   /* llegir dades del fitxer  */
        fprintf(stderr,"Traza 3\n");
        exit(3);}   /* aborta si hi ha algun problema en el fitxer */

    fprintf(stderr,"Traza 4\n");

    fprintf(stderr,"Traza 5\n");

    if (n_args == 3) {  /* si s'ha especificat parametre de retard */
        retard = atoi(ll_args[2]);  /* convertir-lo a enter */
        if (retard < 10)
            retard = 10;    /* verificar limits */
        if (retard > 1000)
            retard = 1000;
    } else
        retard = 100;   /* altrament, fixar retard per defecte */

    printf("Joc del Mur: prem RETURN per continuar:\n");

    //getchar();
    int index = 0;


    if (inicialitza_joc() != 0) /* intenta crear el taulell de joc */
        {
        fprintf(stderr,"Joc iniciat\n");
        exit(4);
        }   /* aborta si hi ha algun problema amb taulell */
    pthread_create(&tid[0],NULL,mou_paleta,(void *) (intptr_t)0);
    //for(int i=1;i<=n_pil;i++)
    pthread_create(&tid[1],NULL,mou_pilota,(void *) (intptr_t)index);

    for(int j=0;j<=n_pil;j++)
        pthread_join(tid[j],NULL);
    fclose(file);
/********** bucle principal del joc **********/
    do {
```

```
        /*if(secs>59){
            if(mins<60){
                mins++;
            }
            secs=0;
        }*/
        /*
        win_retard(retard); // retard del joc
        */
        //secs++;
    win_retard(retard);
    } while (!fi1 && !fi2);
    fprintf(stderr,"Traza 6\n");

    win_fi();       /* tanca les curses */
    fprintf(stderr,"Traza Final\n");

    if (nblocs == 0)
        mostra_final("YOU WIN !");
    else
        mostra_final("GAME OVER");

    pthread_mutex_destroy(&mutex);

    return (0);     /* retorna sense errors d'execucio */
}
```

mur2:

```
/*****************************************************************************/
/*                                                                           */
/*                          mur0.c                                           */
/*                                                                           */
/*  Programa inicial d'exemple per a les practiques 2 de FSO.                */
/*                                                                           */
/*  Compilar i executar:                                                     */
/*     El programa invoca les funcions definides a "winsuport.c", les        */
/*     quals proporcionen una interficie senzilla per crear una finestra     */
/*     de text on es poden escriure caracters en posicions especifiques de   */
/*     la pantalla (basada en CURSES); per tant, el programa necessita ser   */
/*     compilat amb la llibreria 'curses':                                   */
/*                                                                           */
/*       $ gcc -Wall -c winsuport.c -o winsuport.o                           */
/*       $ gcc -Wall mur0.c winsuport.o -o mur0 -lcurses                     */
/*                                                                           */
/*  Al tenir una orientació vertical cal tenir un terminal amb prou files.   */
/*  Per exemple:                                                             */
/*               xterm -geometry 80x50                                       */
/*               gnome-terminal --geometry 80x50                             */
```

```c
/*                                                                      */
/************************************************************************/

//#include <stdint.h>       /* intptr_t for 64bits machines */
#define _REENTRANT
#include <pthread.h>
#include <stdio.h>      /* incloure definicions de funcions estandard */
#include <stdlib.h>
#include <string.h>
#include "winsuport.h"      /* incloure definicions de funcions propies */
#include <stdint.h>
#include <time.h>

/* definicio de constants */
#define MAX_THREADS 10
#define LENGTH 1
#define MAXBALLS 9
#define MIN_FIL 10      /* dimensions del camp. Depenen del terminal ! */
#define MAX_FIL 50
#define MIN_COL 10
#define MAX_COL 80
#define BLKSIZE 3
#define BLKGAP  2
#define BLKCHAR 'B'
#define WLLCHAR '#'
#define FRNTCHAR 'A'
#define LONGMISS    65
#define MAX_THREADS 10
pthread_t tid[MAX_THREADS];
pthread_mutex_t mutex= PTHREAD_MUTEX_INITIALIZER;
            /* variables globals */
char *descripcio[] = {
    "\n",
    "Aquest programa implementa una versio basica del joc Arkanoid o Breakout:\n",
    "generar un camp de joc rectangular amb una porteria, una paleta que s\'ha\n",
    "de moure amb el teclat per a cobrir la porteria, i una pilota que rebota\n",
    "contra les parets del camp, a la paleta i els blocs. El programa acaba si\n",
    "la pilota surt per la porteria o no queden mes blocs. Tambe es pot acabar\n",
    "amb la tecla RETURN.\n",
    "\n",
    "  Arguments del programa:\n",
    "\n",
    "       $ ./mur0 fitxer_config [retard]\n",
    "\n",
    "     El primer argument ha de ser el nom d\'un fitxer de text amb la\n",
    "     configuracio de la partida, on la primera fila inclou informacio\n",
    "     del camp de joc (valors enters), i la segona fila indica posicio\n",
    "     i velocitat de la pilota (valors reals):\n",
    "          num_files  num_columnes  mida_porteria\n",
    "          pos_fil_pal pos_col_pal mida_pal\n"
    "          pos_fila   pos_columna   vel_fila  vel_columna\n",
```

```
    "\n",
    "     on els valors minims i maxims admesos son els seguents:\n",
    "          MIN_FIL <= num_files     <= MAX_FIL\n",
    "          MIN_COL <= num_columnes  <= MAX_COL\n",
    "          0 < mida_porteria < num_files-2\n",
    "        1.0 <= pos_fila     <= num_files-3\n",
    "        1.0 <= pos_columna  <= num_columnes\n",
    "       -1.0 <= vel_fila     <= 1.0\n",
    "       -1.0 <= vel_columna  <= 1.0\n",
    "\n",
    "     Alternativament, es pot donar el valor 0 a num_files i num_columnes\n",
    "     per especificar que es vol que el tauler ocupi tota la pantalla. Si\n",
    "     tambe fixem mida_porteria a 0, el programa ajustara la mida
d\'aquesta\n",
    "     a 3/4 de l\'altura del camp de joc.\n",
    "\n",
    "     A mes, es pot afegir un segon argument opcional per indicar el\n",
    "     retard de moviment del joc en mil.lisegons; el valor minim es 10,\n",
    "     el valor maxim es 1000, i el valor per defecte d'aquest parametre\n",
    "     es 100 (1 decima de segon).\n",
    "\n",
    "  Codis de retorn:\n",
    "     El programa retorna algun dels seguents codis:\n",
    "    0  ==>  funcionament normal\n",
    "    1  ==>  numero d'arguments incorrecte\n",
    "    2  ==>  no s\'ha pogut obrir el fitxer de configuracio\n",
    "    3  ==>  algun parametre del fitxer de configuracio es erroni\n",
    "    4  ==>  no s\'ha pogut crear el camp de joc (no pot iniciar CURSES)\n",
    "\n",
    "    Per a que pugui funcionar aquest programa cal tenir instal.lada la\n",
    "    llibreria de CURSES (qualsevol versio).\n",
    "\n",
    "*"
};              /* final de la descripcio */

int n_fil, n_col;       /* numero de files i columnes del taulell */
int m_por;          /* mida de la porteria (en caracters) */
int f_pal, c_pal;       /* posicio del primer caracter de la paleta */
int m_pal;              /* mida de la paleta */
int f_pil[MAXBALLS], c_pil[MAXBALLS];       /* posicio de la pilota, en valor enter
*/
float pos_f[MAXBALLS], pos_c[MAXBALLS];     /* posicio de la pilota, en valor real
*/
float vel_f[MAXBALLS], vel_c[MAXBALLS];     /* velocitat de la pilota, en valor
real */
int nblocs = 0;
int dirPaleta = 0;
int retard;         /* valor del retard de moviment, en mil.lisegons */
int fi1,fi2;
char strin[LONGMISS];           /* variable per a generar missatges de text */
int n_pil;
```

```
int fi1=0,fi2=0;
int ind=0;
time_t temps_actual,temps_init;
/* funcio per carregar i interpretar el fitxer de configuracio de la partida */
/* el parametre ha de ser un punter a fitxer de text, posicionat al principi */
/* la funcio tanca el fitxer, i retorna diferent de zero si hi ha problemes  */
int carrega_configuracio(FILE * fit)
{
    fprintf(stderr,"Entro\n");
    int ret = 0;
    int i=0;//j=0;
    fscanf(fit, "%d %d %d\n", &n_fil, &n_col, &m_por);  /* camp de joc */
    fprintf(stderr,"n_fil= %d, n_col= %d, m_por= %d\n",n_fil,n_col,m_por);
    fscanf(fit, "%d %d %d\n", &f_pal, &c_pal, &m_pal);  /* paleta */
    fprintf(stderr,"Pilotes =%d\n",n_pil);
    while(!feof(fit)&&i<MAXBALLS){//||(ret<=5&&ret>=0)){
        fprintf(stderr,"Entro bucle %d\n",i);
        fscanf(fit, "%f %f %f %f\n", &pos_f[i], &pos_c[i], &vel_f[i],
&vel_c[i]);   /* pilota */

        //i++;

    if ((n_fil != 0) || (n_col != 0)) { /* si no dimensions maximes */
        if ((n_fil < MIN_FIL) || (n_fil > MAX_FIL) || (n_col < MIN_COL) || (n_col >
MAX_COL))
            ret = 1;
        else
        if (m_por > n_col - 3)
            ret = 2;
        else if ((m_pal > n_col - 3) || (m_pal < 1) || (f_pal > n_fil-1) || (f_pal
< 1) || (c_pal + m_pal > n_col -1 || c_pal < 1 )){
            ret = 3;
            fprintf(stderr,"Traza paso 3:\n");}
        //}
        //while(i<n_pil && ret != 3){
        else if ((pos_f[i] < 1) || (pos_f[i] >= n_fil - 3) || (pos_c[i] < 1)
             || (pos_c[i] > n_col - 1)) /* tres files especials: línia d'estat,
porteria i paleta */
            ret = 4;
    //}
    }
    fprintf(stderr,"Mig\n");
    if ((vel_f[i] < -1.0) || (vel_f[i] > 1.0) || (vel_c[i] < -1.0) || (vel_c[i] >
1.0))
        ret = 5;

    //i++;
    n_pil++;i++;
    }
    fprintf(stderr,"N_pil= %d\n",n_pil);
```

```
    if (ret != 0) {     /* si ha detectat algun error */
        fprintf(stderr, "Error en fitxer de configuracio:\n");
        switch (ret) {
        case 1:
            fprintf(stderr,
                "\tdimensions del camp de joc incorrectes:\n");
            fprintf(stderr, "\tn_fil= %d \tn_col= %d\n", n_fil,
                n_col);
            break;
        case 2:
            fprintf(stderr, "\tmida de la porteria incorrecta:\n");
            fprintf(stderr, "\tm_por= %d\n", m_por);
            break;
        case 3:
            fprintf(stderr,"\tmida de la paleta incorrecta:\n");
            fprintf(stderr, "\tf_pal= %d \tc_pal= %d \t m_pal=%d\n",
f_pal,c_pal,m_pal);
            break;
        case 4:
            fprintf(stderr, "\tposicio de la pilota incorrecta:\n");
            fprintf(stderr, "\tpos_f= %.2f \tpos_c= %.2f\n", pos_f[i], pos_c[i]);
            break;
        case 5:
            fprintf(stderr,
                "\tvelocitat de la pilota incorrecta:\n");
            fprintf(stderr, "\tvel_f= %.2f \tvel_c= %.2f\n", vel_f[i],
                vel_c[i]);
            break;
        }
    }


    fclose(fit);
    fprintf(stderr,"Surto\n");
    return (ret);
}

/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
/* retorna diferent de zero si hi ha algun problema */
int inicialitza_joc(void)
{
    int i, retwin;
    int i_port, f_port; /* inici i final de porteria */
    int c, nb, offset;

    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);
    }

    if (m_por > n_col - 2)
        m_por = n_col - 2;  /* limita valor de la porteria */
    if (m_por == 0)
        m_por = 3 * (n_col - 2) / 4;    /* valor porteria per defecte */

    i_port = n_col / 2 - m_por / 2 - 1; /* crea el forat de la porteria */
    f_port = i_port + m_por - 1;
    for (i = i_port; i <= f_port; i++){
        win_escricar(n_fil - 2, i, ' ', NO_INV);

    }

    n_fil = n_fil - 1;  /* descompta la fila de missatges */

    for (i = 0; i < m_pal; i++) /* dibuixar paleta inicial */{
        win_escricar(f_pal, c_pal + i, '0', INVERS);

    }
    for(i=0;i<n_pil;i++){
    /* generar la pilota */
    if (pos_f[i] > n_fil - 1)
        pos_f[i] = n_fil - 1;   /* limita posicio inicial de la pilota */
    if (pos_c[i] > n_col - 1)
        pos_c[i] = n_col - 1;
    f_pil[i] = pos_f[i];
    c_pil[i] = pos_c[i];        /* dibuixar la pilota inicialment */
    //win_escricar(f_pil[i], c_pil[i], '1', INVERS);

    }
    // generar els blocs
    nb = 0;
    nblocs = n_col / (BLKSIZE + BLKGAP) - 1;
    offset = (n_col - nblocs * (BLKSIZE + BLKGAP) + BLKGAP) / 2;
    for (i = 0; i < nblocs; i++) {
```

```
        for (c = 0; c < BLKSIZE; c++) {
            win_escricar(3, offset + c, FRNTCHAR, INVERS);
            nb++;
            win_escricar(4, offset + c, BLKCHAR, NO_INV);
            nb++;
            win_escricar(5, offset + c, FRNTCHAR, INVERS);
            nb++;
        }
        offset += BLKSIZE + BLKGAP;
    }
    nblocs = nb / BLKSIZE;
    // generar les defenses
    nb = n_col / (BLKSIZE + 2 * BLKGAP) - 2;
    offset = (n_col - nb * (BLKSIZE + 2 * BLKGAP) + BLKGAP) / 2;
    for (i = 0; i < nb; i++) {
        for (c = 0; c < BLKSIZE + BLKGAP; c++) {
            win_escricar(6, offset + c, WLLCHAR, NO_INV);
        }
        offset += BLKSIZE + 2 * BLKGAP;
    }

    sprintf(strin,
        "Tecles: \'%c\'-> Esquerra, \'%c\'-> Dreta, RETURN-> sortir\n",
        TEC_ESQUER, TEC_DRETA);
    win_escristr(strin);
    return (0);
}

/* funcio que escriu un missatge a la línia d'estat i tanca les curses */
void mostra_final(char *miss)
{   int lmarge;
    char marge[LONGMISS];

    /* centrar el misssatge */
    lmarge=(n_col+strlen(miss))/2;
    sprintf(marge,"%%%ds",lmarge);

    sprintf(strin, marge,miss);
    win_escristr(strin);

    /* espera tecla per a que es pugui veure el missatge */
    getchar();
}

/* Si hi ha una col.lisió pilota-bloci esborra el bloc */
void comprovar_bloc(int f, int c)
{
    int col;
    char quin = win_quincar(f, c);
    int pil= ind;
    if (quin == BLKCHAR || quin == FRNTCHAR) {
```

```
        col = c;
        while (win_quincar(f, col) != ' ') {
            win_escricar(f, col, ' ', NO_INV);
            col++;
        }
        col = c - 1;
        while (win_quincar(f, col) != ' ') {
            win_escricar(f, col, ' ', NO_INV);
            col--;
        }

        /* generar nova pilota ? */
        if(quin==BLKCHAR&&ind<=n_pil){
            fprintf(stderr,"Comprovar bloc, n_pil= %d, pil= %d\n",n_pil,pil);
            ind++;
            pthread_create(&tid[ind+1],NULL,mou_pilota,(void *)(intptr_t)pil);
            //pthread_create(&tid[0],NULL,mou_paleta,(void *) (intptr_t)0);
            win_escricar(f_pil[pil],c_pil[pil],'1',INVERS);
        }
        nblocs--;
    }
}



float control_impacte2(int c_pil, float velc0) {
    int distApal;
    float vel_c;

    distApal = c_pil - c_pal;
    if (distApal >= 2*m_pal/3)  /* costat dreta */
        vel_c = 0.5;
    else if (distApal <= m_pal/3)   /* costat esquerra */
        vel_c = -0.5;
    else if (distApal == m_pal/2)   /* al centre */
        vel_c = 0.0;
    else /*: rebot normal */
        vel_c = velc0;
    return vel_c;
}

/* funcio per moure la pilota: retorna un 1 si la pilota surt per la porteria,*/
/* altrament retorna un 0 */
void * mou_pilota(void * index)
{

    int f_h, c_h;
    int pil= (intptr_t) index;
    char rh, rv, rd;
    int fora = 0;

    do{
```

```
    fora=0;
    f_h = pos_f[pil] + vel_f[pil];  /* posicio hipotetica de la pilota (entera) */
    c_h = pos_c[pil] + vel_c[pil];
    rh = rv = rd = ' ';
    if ((f_h != f_pil[pil]) || (c_h != c_pil[pil])) {
    /* si posicio hipotetica no coincideix amb la posicio actual */
        if (f_h != f_pil[pil]) {    /* provar rebot vertical */
            pthread_mutex_lock(&mutex);
            rv = win_quincar(f_h, c_pil[pil]);  /* veure si hi ha algun obstacle */

            if (rv != ' ') {    /* si hi ha alguna cosa */
                comprovar_bloc(f_h, c_pil[pil]);
                pthread_mutex_unlock(&mutex);
                if (rv == '0')  /* col.lisió amb la paleta? */
//                  control_impacte();
                    vel_c[pil] = control_impacte2(c_pil[pil], vel_c[pil]);
                vel_f[pil] = -vel_f[pil];   /* canvia sentit velocitat vertical */
                f_h = pos_f[pil] + vel_f[pil];  /* actualitza posicio hipotetica */
            }
            else{
                pthread_mutex_unlock(&mutex);
            }
        }
        if (c_h != c_pil[pil]) {    /* provar rebot horitzontal */
            pthread_mutex_lock(&mutex);
            rh = win_quincar(f_pil[pil], c_h);  /* veure si hi ha algun obstacle */
            if (rh != ' ') {    /* si hi ha algun obstacle */
                comprovar_bloc(f_pil[pil], c_h);
                pthread_mutex_unlock(&mutex);
                /* TODO?: tractar la col.lisio lateral amb la paleta */
                vel_c[pil] = -vel_c[pil];   /* canvia sentit vel. horitzontal */
                c_h = pos_c[pil] + vel_c[pil];  /* actualitza posicio hipotetica */
            }
            else{
                pthread_mutex_unlock(&mutex);
            }
        }
        if ((f_h != f_pil[pil]) && (c_h != c_pil[pil])) {   /* provar rebot
diagonal */
            pthread_mutex_lock(&mutex); //tanquem semafor
            rd = win_quincar(f_h, c_h);
            if (rd != ' ') {    /* si hi ha obstacle */
                comprovar_bloc(f_h, c_h);
                pthread_mutex_unlock(&mutex);
                vel_f[pil] = -vel_f[pil];
                vel_c[pil] = -vel_c[pil];   /* canvia sentit velocitats */
                f_h = pos_f[pil] + vel_f[pil];
                c_h = pos_c[pil] + vel_c[pil];  /* actualitza posicio entera */
            }
            else{
                pthread_mutex_unlock(&mutex);
```

```
            }
        }
        /* mostrar la pilota a la nova posició */
        pthread_mutex_lock(&mutex);
        if (win_quincar(f_h, c_h) == ' ') { /* verificar posicio definitiva *//* si
no hi ha obstacle */
            //pthread_mutex_lock(&mutex);
            win_escricar(f_pil[pil], c_pil[pil], ' ', NO_INV);  /* esborra pilota
*/
            pthread_mutex_unlock(&mutex);
            pos_f[pil] += vel_f[pil];
            pos_c[pil] += vel_c[pil];
            f_pil[pil] = f_h;
            c_pil[pil] = c_h;   /* actualitza posicio actual */
            if (f_pil[pil] != (n_fil - 1)){ /* si no surt del taulell, */
                pthread_mutex_lock(&mutex);
                win_escricar(f_pil[pil], c_pil[pil], '1', INVERS);  /* imprimeix
pilota */
                pthread_mutex_unlock(&mutex);
            }else{
                pthread_mutex_lock(&mutex);
                fi2 = 1;fora=1;
                pthread_mutex_unlock(&mutex);
                }
        }
        else pthread_mutex_unlock(&mutex);


    } else {    /* posicio hipotetica = a la real: moure */
        pos_f[pil] += vel_f[pil];
        pos_c[pil] += vel_c[pil];
    }
    if((nblocs==0)||(fora==1)){
        pthread_mutex_lock(&mutex);
        fi2=1;
        pthread_mutex_unlock(&mutex);
    }
    win_retard(retard);
    } while (!fi1 && !fi2);
    return ((void *) index);
}

/* funcio per moure la paleta segons la tecla premuda */
/* retorna un boolea indicant si l'usuari vol acabar */

void * mou_paleta(void * nul)
{
    //int tecla, result;
    int tecla;

do{
```

```
    //result = 0;
    pthread_mutex_lock(&mutex);
    tecla = win_gettec();
    pthread_mutex_unlock(&mutex);
    if (tecla != 0) {
        if ((tecla == TEC_DRETA) && ((c_pal + m_pal) < n_col - 1)) {
                pthread_mutex_lock(&mutex);
                win_escricar(f_pal, c_pal, ' ', NO_INV);    /* esborra primer bloc
*/
                //pthread_mutex_unlock(&mutex);
                c_pal++;    /* actualitza posicio */
                //pthread_mutex_lock(&mutex);
                win_escricar(f_pal, c_pal + m_pal - 1, '0', INVERS);    /*esc.
ultim bloc */
                pthread_mutex_unlock(&mutex);
        }
        if ((tecla == TEC_ESQUER) && (c_pal > 1)) {
                pthread_mutex_lock(&mutex);
                win_escricar(f_pal, c_pal + m_pal - 1, ' ', NO_INV);    /*esborra
ultim bloc */
                pthread_mutex_unlock(&mutex);
                c_pal--;    /* actualitza posicio */
                pthread_mutex_lock(&mutex);
                win_escricar(f_pal, c_pal, '0', INVERS);    /* escriure primer bloc
*/

                pthread_mutex_unlock(&mutex);
        }
        if (tecla == TEC_RETURN){
            pthread_mutex_lock(&mutex);
            fi1 = 1;
            pthread_mutex_unlock(&mutex);
            }
                /* final per pulsacio RETURN */
        dirPaleta = tecla;  /* per a afectar al moviment de les pilotes */
    }
    time(&temps_actual);
    win_retard(retard);
    double diferencia=difftime(temps_actual,temps_init);
    int minute=(int)diferencia/60;
    int seconds=(int)diferencia-minute*60;
    char temps[20];
    sprintf(temps,"%d : %d",minute,seconds);
    pthread_mutex_lock(&mutex);
    win_escristr(temps);
    pthread_mutex_unlock(&mutex);
    }while (!fi1 && !fi2);
    return ((void*) 0);
}

/* programa principal */
```

```c
int main(int n_args, char *ll_args[])
{
    time(&temps_init);
    FILE *file;
    file=fopen("traza.txt","w");

    int i;
    //n_pil=1;
    FILE *fit_conf;

    fprintf(stderr,"Traza 1\n");

    if ((n_args != 2) && (n_args != 3)) {   /* si numero d'arguments incorrecte */
        i = 0;
        do
            fprintf(stderr, "%s", descripcio[i++]); /* imprimeix descripcio */
        while (descripcio[i][0] != '*');    /* mentre no arribi al final */
        exit(1);
    }

    fit_conf = fopen(ll_args[1], "rt"); /* intenta obrir el fitxer */

    if (!fit_conf) {
        fprintf(stderr, "Error: no s'ha pogut obrir el fitxer \'%s\'\n",
            ll_args[1]);
        exit(2);
    }

    fprintf(stderr,"Traza 2\n");

    if (carrega_configuracio(fit_conf) != 0){   /* llegir dades del fitxer  */
        //fprintf(file,"Traza 3\n");
        fprintf(stderr,"Traza 3\n");
        exit(3);}   /* aborta si hi ha algun problema en el fitxer */

    fprintf(stderr,"Traza 4\n");

    fprintf(stderr,"Traza 5\n");

    if (n_args == 3) {  /* si s'ha especificat parametre de retard */
        retard = atoi(ll_args[2]);  /* convertir-lo a enter */
        if (retard < 10)
            retard = 10;    /* verificar limits */
        if (retard > 1000)
            retard = 1000;
    } else
        retard = 100;   /* altrament, fixar retard per defecte */


    printf("Joc del Mur: prem RETURN per continuar:\n");
```

```
    //getchar();
    int index = 0;

    if (inicialitza_joc() != 0) /* intenta crear el taulell de joc */
        {
        fprintf(stderr,"Joc iniciat\n");
        exit(4);
        }   /* aborta si hi ha algun problema amb taulell */
    pthread_create(&tid[0],NULL,mou_paleta,(void *) (intptr_t)0);
    //for(int i=1;i<=n_pil;i++)
    pthread_create(&tid[1],NULL,mou_pilota,(void *) (intptr_t)index);

    for(int j=0;j<=1;j++)
        pthread_join(tid[j],NULL);
    fclose(file);
/********** bucle principal del joc **********/
    do {
        /*if(secs>59){
            if(mins<60){
                mins++;
            }
            secs=0;
        }*/
        /*fi1 = mou_paleta();
        fi2=0;
        fi2 = mou_pilota();
        win_retard(retard); // retard del joc
        */
        //secs++;
    win_retard(retard);
    } while (!fi1 && !fi2);
    fprintf(stderr,"Traza 6\n");

    win_fi();       /* tanca les curses */
    fprintf(stderr,"Traza Final\n");

    if (nblocs == 0)
        mostra_final("YOU WIN !");
    else
        mostra_final("GAME OVER");

    pthread_mutex_destroy(&mutex);

    return (0);     /* retorna sense errors d'execucio */
}
```

Conclusions i Autoavaluació:

Com a conclusió, tot i que havent-li dedicat moltes hores d'esforç no hem aconseguit el nostre objectiu que era fer funcionar totes les fases, considerem que l'aprenentatge duut a terme en aquesta pràctica ha sigut molt elevat. Ens hem topat amb un gran número de problemes que hem hagut de anar solucionant com hem pogut i amb d'altres que encara no sabem com resoldre. Tot i això no ens hem quedat estancats en una fase i hem seguit fins al final per no deixar res i apendre el màxim possible.
Hi han hagut numerosos moments de frustració durant el desenvolupament de la pràctica però no ens hem donat per vençuts. Considerem que hem fet una feina honesta i que l'objectiu que era apendre, ha estat assolit.

Autoavaluació: **5**