Pràctica 2

# Pràctica 2.1

## Fase 1:

### Disseny

Per a dissenyar la nostra pràctica hem modificat el codi proporcionat pels professors de l’assignatura de tal manera que tinguem 2 funcions separades:

*mou\_menjacocos()*

*mou\_fantasma()*

La nostra és idea és crear un thread que controli el menjacocos i un altre per cada fantasma. Aquests threads aniran controlant els seus elements fins que s’aturi la execució del joc.

El joc es pot aturar quan algun fantasma menja el menjacocos o quan l’usuari ha aconseguit menjar tots els cocos, això es controla amb les variables *fi1* i *fi2*, quan alguna d’aquestes variables pren el valor de cert s’aturen tots els threads.

Per acabar hem plantejat el codi principal de tal manera que simplement creï els threads i esperi a que finalitzin per després analitzar el resultat que han retornat.

### Implementació

A continuació s’adjunta la solució proposada:

#include <stdio.h> /\* incloure definicions de funcions estandard \*/

#include <stdlib.h> /\* per exit() \*/

#include <unistd.h> /\* per getpid() \*/

#include "winsuport.h" /\* incloure definicions de funcions propies \*/

#include <pthread.h>

#define MIN\_FIL 7 /\* definir limits de variables globals \*/

#define MAX\_FIL 25

#define MIN\_COL 10

#define MAX\_COL 80

#define MAX\_THREADS 10

#define MAX\_FANTASMES 100

/\* definir estructures d'informacio \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

/\* variables globals \*/

int n\_fil1, n\_col; /\* dimensions del camp de joc \*/

char tauler[70]; /\* nom del fitxer amb el laberint de joc 70\*/

char c\_req; /\* caracter de pared del laberint \*/

pthread\_t tid[MAX\_THREADS];

pthread\_t coco;

objecte mc; /\* informacio del menjacocos \*/

objecte fantasmes[10]; /\* informacio del fantasma 1 \*/

int n\_fantasmes = 0;

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

int cocos; /\* numero restant de cocos per menjar \*/

int retard; /\* valor del retard de moviment, en mil.lisegons \*/

int fi1=0, fi2=0;

/\* funcio per realitzar la carrega dels parametres de joc emmagatzemats \*/

/\* dins d'un fitxer de text, el nom del qual es passa per referencia a \*/

/\* '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 al 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 %s %c\n",&n\_fil1,&n\_col,tauler,&c\_req);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((n\_fil1 < MIN\_FIL) || (n\_fil1 > MAX\_FIL) ||

(n\_col < MIN\_COL) || (n\_col > MAX\_COL))

{

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

fprintf(stderr,"\t%d =< n\_fil1 (%d) =< %d\n",MIN\_FIL,n\_fil1,MAX\_FIL);

fprintf(stderr,"\t%d =< n\_col (%d) =< %d\n",MIN\_COL,n\_col,MAX\_COL);

fclose(fit);

exit(3);

}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((mc.f < 1) || (mc.f > n\_fil1-3) ||

(mc.c < 1) || (mc.c > n\_col-2) ||

(mc.d < 0) || (mc.d > 3))

{

fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");

fprintf(stderr,"\t1 =< mc.f (%d) =< n\_fil1-3 (%d)\n",mc.f,(n\_fil1-3));

fprintf(stderr,"\t1 =< mc.c (%d) =< n\_col-2 (%d)\n",mc.c,(n\_col-2));

fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);

fclose(fit);

exit(4);

}

int i = 0;

while(!feof(fit)){

fscanf(fit,"%d %d %d %f\n",&fantasmes[i].f,&fantasmes[i].c,&fantasmes[i].d,&fantasmes[i].r);

if ((fantasmes[i].f < 1) || (fantasmes[i].f > n\_fil1-3) ||

(fantasmes[i].c < 1) || (fantasmes[i].c > n\_col-2) ||

(fantasmes[i].d < 0) || (fantasmes[i].d > 3))

{

fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");

fprintf(stderr,"\t1 =< f1.f (%d) =< n\_fil1-3 (%d)\n",fantasmes[i].f,(n\_fil1-3));

fprintf(stderr,"\t1 =< f1.c (%d) =< n\_col-2 (%d)\n",fantasmes[i].c,(n\_col-2));

fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",fantasmes[i].d);

fclose(fit);

exit(5);

}else{

fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",i,fantasmes[i].f,fantasmes[i].c);

}

i++;

}

n\_fantasmes = i;

fprintf(stderr,"El valor del numero de fantasmes és %d\n",i);

fclose(fit); /\* fitxer carregat: tot OK! \*/

printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",

TEC\_AMUNT, TEC\_AVALL, TEC\_DRETA, TEC\_ESQUER);

printf("prem una tecla per continuar:\n");

getchar();

}

/\* funcio per moure un fantasma una posicio; retorna 1 si el fantasma \*/

/\* captura al menjacocos, 0 altrament \*/

//int mou\_fantasma(objecte f1)

void \* mou\_fantasma(void \*index)

{

int i = (intptr\_t) index;

objecte f1 = fantasmes[i];

if(i < 0){

fprintf(stderr,"Error: index incorrecte\n");

exit(6);

}

objecte seg;

int ret;

int k, vk, nd, vd[3];

ret = 0;

do{

nd = 0; //numero de direccions disponibles

for (k=-1; k<=1; k++) /\* provar direccio actual i dir. veines \*/

{

vk = (f1.d + k) % 4; /\* direccio veina \*/

if (vk < 0) vk += 4; /\* corregeix negatius \*/

seg.f = f1.f + df[vk]; /\* calcular posicio en la nova dir.\*/

seg.c = f1.c + dc[vk];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0'))

{

vd[nd] = vk; /\* memoritza com a direccio possible \*/

nd++;

}

}

if (nd == 0){ /\* si no pot continuar, \*/

f1.d = (f1.d + 2) % 4; /\* canvia totalment de sentit \*/

}else

{

if (nd == 1){ /\* si nomes pot en una direccio \*/

f1.d = vd[0]; /\* li assigna aquesta \*/

}

else{ /\* altrament \*/

f1.d = vd[rand() % nd]; /\* segueix una dir. aleatoria \*/

}

seg.f = f1.f + df[f1.d]; /\* calcular seguent posicio final \*/

seg.c = f1.c + dc[f1.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

win\_escricar(f1.f,f1.c,f1.a,NO\_INV); /\* esborra posicio anterior \*/

f1.f = seg.f;

f1.c = seg.c;

f1.a = seg.a; /\* actualitza posicio \*/

win\_escricar(f1.f,f1.c,'1'+i,NO\_INV); /\* redibuixa fantasma \*/

if (f1.a == '0') ret = 1; /\* ha capturat menjacocos \*/

}

fi2=ret;

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi2);

}

/\* funcio per moure el menjacocos una posicio, en funcio de la direccio de \*/

/\* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots \*/

/\* els cocos, i 0 altrament \*/

//int mou\_menjacocos(void)

void \* mou\_menjacocos(void \* null)

{

char strin[12];

objecte seg;

int tec, ret;

ret = 0;

do{

tec = win\_gettec();

if (tec != 0)

switch (tec) /\* modificar direccio menjacocos segons tecla \*/

{

case TEC\_AMUNT: mc.d = 0; break;

case TEC\_ESQUER: mc.d = 1; break;

case TEC\_AVALL: mc.d = 2; break;

case TEC\_DRETA: mc.d = 3; break;

case TEC\_RETURN: ret = -1; break;

}

seg.f = mc.f + df[mc.d]; /\* calcular seguent posicio \*/

seg.c = mc.c + dc[mc.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a == ' ') || (seg.a == '.'))

{

win\_escricar(mc.f,mc.c,' ',NO\_INV); /\* esborra posicio anterior \*/

mc.f = seg.f; mc.c = seg.c; /\* actualitza posicio \*/

win\_escricar(mc.f,mc.c,'0',NO\_INV); /\* redibuixa menjacocos \*/

if (seg.a == '.')

{

cocos--;

sprintf(strin,"Cocos: %d", cocos); win\_escristr(strin);

//sprintf(strin,"Fanta: %d",n\_fantasmes); win\_escristr(strin);

if (cocos == 0) ret = 1;

}

}

fi1=ret;

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi1);

}

/\* funcio per inicialitar les variables i visualitzar l'estat inicial del joc \*/

void inicialitza\_joc(void)

{

int r,i,j;

char strin[12];

int ex\_code = 0;

r = win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

if (r == 0)

{

mc.a = win\_quincar(mc.f,mc.c);

if (mc.a == c\_req) r = -6; /\* error: menjacocos sobre pared \*/

else

{

for(int i = 0; i < n\_fantasmes; i++){

fantasmes[i].a = win\_quincar(fantasmes[i].f,fantasmes[i].c);

if (fantasmes[i].a == c\_req){

r = -7; /\* error: fantasma sobre pared \*/

ex\_code = i;

}

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

}

}

if(r==0){

cocos = 0; /\* compta el numero total de cocos \*/

for (i=0; i<n\_fil1-1; i++)

for (j=0; j<n\_col; j++){

if (win\_quincar(i,j)=='.') cocos++;

}

win\_escricar(mc.f,mc.c,'0',NO\_INV);

if (mc.a == '.') cocos--; /\* menja primer coco \*/

sprintf(strin,"Cocos: %d", cocos);

win\_escristr(strin);

}

if (r != 0)

{ win\_fi();

fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");

switch (r)

{ case -1: fprintf(stderr," nom de fitxer erroni\n"); break;

case -2: fprintf(stderr," numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;

case -3: fprintf(stderr," numero de columnes del laberint incorrecte\n"); break;

case -4: fprintf(stderr," numero de files del laberint incorrecte\n"); break;

case -5: fprintf(stderr," finestra de camp de joc no oberta\n"); break;

case -6: fprintf(stderr," posicio inicial del menjacocos damunt la pared del laberint\n"); break;

case -7: fprintf(stderr," posicio inicial del fantasma %d damunt la pared del laberint\n",ex\_code); break;

}

exit(7);

}

}

void crear\_threads(){

int i;

if(pthread\_create(&coco,NULL,mou\_menjacocos,NULL)!=0){

fprintf(stderr,"Error al crear el thread del menjacocos\n");

exit(1);

}

for(i = 0; i < n\_fantasmes; i++){

if(i<MAX\_FANTASMES){

if(pthread\_create(&tid[i],NULL,mou\_fantasma,(void \*)(intptr\_t) i)!=0){

fprintf(stderr,"Error al crear el thread %d\n",i);

exit(1);

}

}

}

}

/\* programa principal \*/

int main(int n\_args, const char \*ll\_args[])

{

int rc; /\* variables locals \*/

srand(getpid()); /\* inicialitza numeros aleatoris \*/

if ((n\_args != 2) && (n\_args !=3))

{ fprintf(stderr,"Comanda: cocos0 fit\_param [retard]\n");

exit(1);

}

carrega\_parametres(ll\_args[1]);

//fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",1,fantasmes[1].f,fantasmes[1].c);

if (n\_args == 3) retard = atoi(ll\_args[2]);

else retard = 100;

rc = win\_ini(&n\_fil1,&n\_col,'+',INVERS); /\* intenta crear taulell \*/

if (rc == 0) /\* si aconsegueix accedir a l'entorn CURSES \*/

{

inicialitza\_joc();

crear\_threads(); //creem els threads

//fi1 = 0; fi2 = 0;

/\*do{

win\_retard(retard);

}while(!fi1 && !fi2);\*/

for(int th=0; th<n\_fantasmes;th++){

pthread\_join(tid[th], (void \*\*)&fi2);

}

pthread\_join(coco, (void \*\*)&fi1);

win\_fi();

if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");

else { if (fi1) printf("Ha guanyat l'usuari!\n");

else printf("Ha guanyat l'ordinador!\n"); }

}

else

{ fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");

switch (rc)

{ 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;

}

exit(6);

}

return(0);

}

### Joc de Proves

Per a comprovar el funcionament del nostre codi s’han fet els següents jocs de proves:

|  |  |  |
| --- | --- | --- |
| **Especificació:** | **Paràmetres** | **Resultat esperat** |
| Prova d’un tauler buit sense fantasmes, només amb el menjacocos. | * prova1.txt | El jugador pot moure el menjacocos pel mapa, no apareix cap fantasma.  El jugador guanya quan ha menjat tots els cocos. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Direcció fantasma 1 fora de rang | * prova2.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Fila fantasma 1 fora de rang | * prova3.txt | El programa mostra un error i no es pot jugar. |
| Prova de posar un fantasma sobre la paret. | * prova4.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb un nombre de fantasmes més gran que MAX\_FANTASMES. | * prova5.txt | No es permet jugar ja que hi ha una inconsistència en els paràmetres. |
| Prova d’un joc normal | * prova6.txt | Es permet jugar, però es noten errors de sincronització entre els threads fent que el joc sigui injugable. |

## Fase 2:

### Disseny:

Per a fer el disseny d’aquesta part hem analitzat les variables globals que es poden fer servir des de 2 threads concurrentment.

Hem identificat principalment que la secció en la qual garantir la exclusivitat d’accés a la memòria és la part de dins del bucle, així que hem implementat el següent pseudocodi:

*bloquejar()*

*do*

*{*

*[..]*

*desbloquejar()*

*// permetem al següent thread accedir a la memòria*

*bloquejar()*

*} while( .. )*

*desbloquejar()*

La nostra idea es que així tant podem permetre que els diferents threads puguin treballar concurrentment reservant la secció crítica de cada un.

Cal remarcar que aquest esquema s’ha implementat tant per moure els fantasmes com per moure el menjacocos, així garantim una integritat total al codi.

### Implementació:

S’ha realitzat la següent implementació del codi de solució:

#include <stdio.h> /\* incloure definicions de funcions estandard \*/

#include <stdlib.h> /\* per exit() \*/

#include <unistd.h> /\* per getpid() \*/

#include "winsuport.h" /\* incloure definicions de funcions propies \*/

#include <pthread.h>

#define MIN\_FIL 7 /\* definir limits de variables globals \*/

#define MAX\_FIL 25

#define MIN\_COL 10

#define MAX\_COL 80

#define MAX\_THREADS 10

#define MAX\_FANTASMES 100

/\* definir estructures d'informacio \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

/\* variables globals \*/

int n\_fil1, n\_col; /\* dimensions del camp de joc \*/

char tauler[70]; /\* nom del fitxer amb el laberint de joc 70\*/

char c\_req; /\* caracter de pared del laberint \*/

pthread\_t tid[MAX\_THREADS];

pthread\_t coco;

objecte mc; /\* informacio del menjacocos \*/

objecte fantasmes[10]; /\* informacio del fantasma 1 \*/

int n\_fantasmes = 0;

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

int cocos; /\* numero restant de cocos per menjar \*/

int retard; /\* valor del retard de moviment, en mil.lisegons \*/

int fi1=0, fi2=0;

pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;

//pthread\_mutex\_t mutex2=PTHREAD\_MUTEX\_INITIALIZER;

/\* funcio per realitzar la carrega dels parametres de joc emmagatzemats \*/

/\* dins d'un fitxer de text, el nom del qual es passa per referencia a \*/

/\* '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 al 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 %s %c\n",&n\_fil1,&n\_col,tauler,&c\_req);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((n\_fil1 < MIN\_FIL) || (n\_fil1 > MAX\_FIL) ||

(n\_col < MIN\_COL) || (n\_col > MAX\_COL))

{

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

fprintf(stderr,"\t%d =< n\_fil1 (%d) =< %d\n",MIN\_FIL,n\_fil1,MAX\_FIL);

fprintf(stderr,"\t%d =< n\_col (%d) =< %d\n",MIN\_COL,n\_col,MAX\_COL);

fclose(fit);

exit(3);

}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((mc.f < 1) || (mc.f > n\_fil1-3) ||

(mc.c < 1) || (mc.c > n\_col-2) ||

(mc.d < 0) || (mc.d > 3))

{

fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");

fprintf(stderr,"\t1 =< mc.f (%d) =< n\_fil1-3 (%d)\n",mc.f,(n\_fil1-3));

fprintf(stderr,"\t1 =< mc.c (%d) =< n\_col-2 (%d)\n",mc.c,(n\_col-2));

fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);

fclose(fit);

exit(4);

}

int i = 0;

while(!feof(fit)){

fscanf(fit,"%d %d %d %f\n",&fantasmes[i].f,&fantasmes[i].c,&fantasmes[i].d,&fantasmes[i].r);

if ((fantasmes[i].f < 1) || (fantasmes[i].f > n\_fil1-3) ||

(fantasmes[i].c < 1) || (fantasmes[i].c > n\_col-2) ||

(fantasmes[i].d < 0) || (fantasmes[i].d > 3))

{

fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");

fprintf(stderr,"\t1 =< f1.f (%d) =< n\_fil1-3 (%d)\n",fantasmes[i].f,(n\_fil1-3));

fprintf(stderr,"\t1 =< f1.c (%d) =< n\_col-2 (%d)\n",fantasmes[i].c,(n\_col-2));

fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",fantasmes[i].d);

fclose(fit);

exit(5);

}else{

fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",i,fantasmes[i].f,fantasmes[i].c);

}

i++;

}

n\_fantasmes = i;

fprintf(stderr,"El valor del numero de fantasmes és %d\n",i);

fclose(fit); /\* fitxer carregat: tot OK! \*/

printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",

TEC\_AMUNT, TEC\_AVALL, TEC\_DRETA, TEC\_ESQUER);

printf("prem una tecla per continuar:\n");

getchar();

}

/\* funcio per moure un fantasma una posicio; retorna 1 si el fantasma \*/

/\* captura al menjacocos, 0 altrament \*/

//int mou\_fantasma(objecte f1)

void \* mou\_fantasma(void \*index)

{

int i = (intptr\_t) index;

objecte f1 = fantasmes[i];

if(i < 0){

fprintf(stderr,"Error: index incorrecte\n");

exit(6);

}

objecte seg;

int ret;

int k, vk, nd, vd[3];

ret = 0;

pthread\_mutex\_lock(&mutex);

do{

nd = 0; //numero de direccions disponibles

for (k=-1; k<=1; k++) /\* provar direccio actual i dir. veines \*/

{

vk = (f1.d + k) % 4; /\* direccio veina \*/

if (vk < 0) vk += 4; /\* corregeix negatius \*/

seg.f = f1.f + df[vk]; /\* calcular posicio en la nova dir.\*/

seg.c = f1.c + dc[vk];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0'))

{

vd[nd] = vk; /\* memoritza com a direccio possible \*/

nd++;

}

}

if (nd == 0){ /\* si no pot continuar, \*/

f1.d = (f1.d + 2) % 4; /\* canvia totalment de sentit \*/

}else

{

if (nd == 1){ /\* si nomes pot en una direccio \*/

f1.d = vd[0]; /\* li assigna aquesta \*/

}

else{ /\* altrament \*/

f1.d = vd[rand() % nd]; /\* segueix una dir. aleatoria \*/

}

seg.f = f1.f + df[f1.d]; /\* calcular seguent posicio final \*/

seg.c = f1.c + dc[f1.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

win\_escricar(f1.f,f1.c,f1.a,NO\_INV); /\* esborra posicio anterior \*/

f1.f = seg.f;

f1.c = seg.c;

f1.a = seg.a; /\* actualitza posicio \*/

win\_escricar(f1.f,f1.c,'1'+i,NO\_INV); /\* redibuixa fantasma \*/

if (f1.a == '0') ret = 1; /\* ha capturat menjacocos \*/

}

fi2=ret;

pthread\_mutex\_unlock(&mutex);

win\_retard(mc.r\*retard);

pthread\_mutex\_lock(&mutex);

}while(!fi1 && !fi2);

pthread\_mutex\_unlock(&mutex);

return ((void\*) (intptr\_t) fi2);

}

/\* funcio per moure el menjacocos una posicio, en funcio de la direccio de \*/

/\* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots \*/

/\* els cocos, i 0 altrament \*/

//int mou\_menjacocos(void)

void \* mou\_menjacocos(void \* null)

{

char strin[12];

objecte seg;

int tec, ret;

ret = 0;

do{

pthread\_mutex\_lock(&mutex);

tec = win\_gettec();

if (tec != 0)

switch (tec) /\* modificar direccio menjacocos segons tecla \*/

{

case TEC\_AMUNT: mc.d = 0; break;

case TEC\_ESQUER: mc.d = 1; break;

case TEC\_AVALL: mc.d = 2; break;

case TEC\_DRETA: mc.d = 3; break;

case TEC\_RETURN: ret = -1; break;

}

seg.f = mc.f + df[mc.d]; /\* calcular seguent posicio \*/

seg.c = mc.c + dc[mc.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a == ' ') || (seg.a == '.'))

{

win\_escricar(mc.f,mc.c,' ',NO\_INV); /\* esborra posicio anterior \*/

mc.f = seg.f; mc.c = seg.c; /\* actualitza posicio \*/

win\_escricar(mc.f,mc.c,'0',NO\_INV); /\* redibuixa menjacocos \*/

if (seg.a == '.')

{

cocos--;

sprintf(strin,"Cocos: %d", cocos); win\_escristr(strin);

//sprintf(strin,"Fanta: %d",n\_fantasmes); win\_escristr(strin);

if (cocos == 0) ret = 1;

}

}

fi1=ret;

pthread\_mutex\_unlock(&mutex);

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi1);

}

/\* funcio per inicialitar les variables i visualitzar l'estat inicial del joc \*/

void inicialitza\_joc(void)

{

int r,i,j;

char strin[12];

int ex\_code = 0;

r = win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

if (r == 0)

{

mc.a = win\_quincar(mc.f,mc.c);

if (mc.a == c\_req) r = -6; /\* error: menjacocos sobre pared \*/

else

{

for(int i = 0; i < n\_fantasmes; i++){

fantasmes[i].a = win\_quincar(fantasmes[i].f,fantasmes[i].c);

if (fantasmes[i].a == c\_req){

r = -7; /\* error: fantasma sobre pared \*/

ex\_code = i;

}

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

}

}

if(r==0){

cocos = 0; /\* compta el numero total de cocos \*/

for (i=0; i<n\_fil1-1; i++)

for (j=0; j<n\_col; j++){

pthread\_mutex\_lock(&mutex);

if (win\_quincar(i,j)=='.') cocos++;

pthread\_mutex\_unlock(&mutex);

}

pthread\_mutex\_lock(&mutex);

win\_escricar(mc.f,mc.c,'0',NO\_INV);

pthread\_mutex\_unlock(&mutex);

pthread\_mutex\_lock(&mutex);

if (mc.a == '.') cocos--; /\* menja primer coco \*/

pthread\_mutex\_unlock(&mutex);

sprintf(strin,"Cocos: %d", cocos);

pthread\_mutex\_lock(&mutex);

win\_escristr(strin);

pthread\_mutex\_unlock(&mutex);

}

if (r != 0)

{ win\_fi();

fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");

switch (r)

{ case -1: fprintf(stderr," nom de fitxer erroni\n"); break;

case -2: fprintf(stderr," numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;

case -3: fprintf(stderr," numero de columnes del laberint incorrecte\n"); break;

case -4: fprintf(stderr," numero de files del laberint incorrecte\n"); break;

case -5: fprintf(stderr," finestra de camp de joc no oberta\n"); break;

case -6: fprintf(stderr," posicio inicial del menjacocos damunt la pared del laberint\n"); break;

case -7: fprintf(stderr," posicio inicial del fantasma %d damunt la pared del laberint\n",ex\_code); break;

}

exit(7);

}

}

void crear\_threads(){

int i;

if(pthread\_create(&coco,NULL,mou\_menjacocos,NULL)!=0){

fprintf(stderr,"Error al crear el thread del menjacocos\n");

exit(1);

}

for(i = 0; i < n\_fantasmes; i++){

if(i<MAX\_FANTASMES){

if(pthread\_create(&tid[i],NULL,mou\_fantasma,(void \*)(intptr\_t) i)!=0){

fprintf(stderr,"Error al crear el thread %d\n",i);

exit(1);

}

}

}

}

/\* programa principal \*/

int main(int n\_args, const char \*ll\_args[])

{

int rc; /\* variables locals \*/

srand(getpid()); /\* inicialitza numeros aleatoris \*/

if ((n\_args != 2) && (n\_args !=3))

{ fprintf(stderr,"Comanda: cocos0 fit\_param [retard]\n");

exit(1);

}

carrega\_parametres(ll\_args[1]);

//fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",1,fantasmes[1].f,fantasmes[1].c);

if (n\_args == 3) retard = atoi(ll\_args[2]);

else retard = 100;

rc = win\_ini(&n\_fil1,&n\_col,'+',INVERS); /\* intenta crear taulell \*/

if (rc == 0) /\* si aconsegueix accedir a l'entorn CURSES \*/

{

pthread\_mutex\_init(&mutex, NULL);

inicialitza\_joc();

crear\_threads(); //creem els threads

//fi1 = 0; fi2 = 0;

/\*do{

win\_retard(retard);

}while(!fi1 && !fi2);\*/

for(int th=0; th<n\_fantasmes;th++){

pthread\_join(tid[th], (void \*\*)&fi2);

}

pthread\_join(coco, (void \*\*)&fi1);

pthread\_mutex\_destroy(&mutex);

win\_fi();

if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");

else { if (fi1) printf("Ha guanyat l'usuari!\n");

else printf("Ha guanyat l'ordinador!\n"); }

}

else

{ fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");

switch (rc)

{ 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;

}

exit(6);

}

return(0);

}

### Joc de proves:

Per a comprovar el funcionament del nostre codi s’han fet els següents jocs de proves:

|  |  |  |
| --- | --- | --- |
| **Especificació:** | **Paràmetres** | **Resultat esperat** |
| Prova d’un tauler buit sense fantasmes, només amb el menjacocos. | * prova1.txt | El jugador pot moure el menjacocos pel mapa, no apareix cap fantasma.  El jugador guanya quan ha menjat tots els cocos. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Direcció fantasma 1 fora de rang | * prova2.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Fila fantasma 1 fora de rang | * prova3.txt | El programa mostra un error i no es pot jugar. |
| Prova de posar un fantasma sobre la paret. | * prova4.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb un nombre de fantasmes més gran que MAX\_FANTASMES. | * prova5.txt | No es permet jugar ja que hi ha una inconsistència en els paràmetres. |
| Prova d’un joc normal | * prova6.txt | Es permet jugar sense problemes i no surten errors de sincronització entre els threads. |