Chicken Little Dodge

PRÀCTICA VIDEOJOC EASY68K

Nadal Comparini Bauzá (43209381V)

Miguel Vidal Coll (43225456S)

ESTRUCTURA DE COMPUTADORS II | UIB

Introducció

A continuació explicarem les diferents parts de la nostra practica de l'assignatura Estructura de computadors II, en la qual hem implementat un videojoc arcade, repicant el disseny que tindria un joc de maquines recreatives, ambientat en la pel·lícula Chicken Little, ja que el nostre videojoc tracta d'esquivar els núvols que cauen del cel i recollir el major nombre de monedes per obtenir la màxima puntuació possible. Hem decidit ambientar el nostre videojoc en una pel·lícula per mostrar a l'usuari una temàtica que pugui reconèixer i per afegir-li un toc d'humor.

Estructura del codi

El videojoc consta de 14 arxius del EASy68K.

1. **7SEG.X68:** Conté les subrutines relacionades amb el visualitzador de 7 segments. Hem aprofitat les subrutines proporcionades pel professor. Tot i que també hem creat noves subrutines.

2. **AGENTLST.X68:** S'encarrega dels agents que crea i destrueix el videojoc. Les barres de col·lisió i les monedes son agents. Es guarden a la memòria dinàmica i en el moment de destruir aquests agents s'allibera de la memòria. Cal destacar que tots els agents segueixen la mateixa estructura de dades.

3. **CLOUD.X68:** Es controla tot el que involucra a l'agent que haurà d'esquivar el jugador. La creació, l'actualització de posició i el pintat per pantalla d'aquest obstacle es troba a aquest arxiu. Cal mencionar que aquí es on es fa la cridada a la subrutina per reproduir els sons relacionats amb aquest agent.

4. **COIN.X68:** Les monedes es controlen de la mateixa manera que l'agent anterior amb un arxiu a part. La creació (amb menor freqüència que les barres),l'actualització de les posicions i el pintat per pantalla es duen a terme a aquest arxiu.

5. **CONST.X68:** Conté una sèrie de constants organitzades segons l'entorn al qual pertanyen. Volem fer especial menció, sobre tot, a les relacionades amb el jugador i amb la música ja que són les

6. **MAIN.X68:** Conté el bucle principal i els *INCLUDE* necessaris per el funcionament del joc.

7. **MUSIC.X68:** A aquest arxiu es dur a terme tot el control de la música mitjançant la creació de subrutines genèriques que es poden emprar

amb diferents sons segons els *inputs*. El controlador de música que s'empra és DirectX de EASy68K.

8. **PLAYER.X68:** De manera similar a com es controlen els agents es controla el jugador. Aquest arxiu inicialitza el jugador, permet l'actualització de la seva posició i el repintat per pantalla. Cal mencionar que encara que el pintat per pantalla sembli caòtic i excessiu, es realitza d'aquesta manera ja que es pinta el jugador per parts.

9. **SPAWNER.X68:** Aquest arxiu que controla la generació d'agents és similar al proporcionat a l'exemple amb una petita modificació per tal de generar també les monedes de manera aleatòria. A més dins aquest arxiu també es pinta el background del joc.

10. **STATES.X68:** S'encarrega de controlar els diferents estats pels quals avança el programa. És molt similar al proporcionat a l'exemple. No obstant, s'han afegit modificacions a les subrutines encarregades de generar els estats inicials i *game over*.

11. **SYSCONST.X68:** Són constants del sistema proporcionades a l'exemple. No han estat modificades.

12. **SYSTEM.X68:** Conté una sèrie de rutines del sistema. Les subrutines SCRINIT, SCRUPD i KBDREAD han estat implementades per nosaltres segons les especificacions indicades als comentaris.

13. **SYSVAR.X68**: Conté una sèrie de variables del sistema. No han estat modificades.

14. **UTIL.X68:** A aquest arxiu es troben un conjunt de macros i subrutines per poder simplificar i clarificar el codi en general. Hem emprat en gran part les proporcionades a l'exemple però també hem afegit algunes i modificat d'altres.

15. **VAR.X68:** Conté una sèrie de variables del joc. Hem afegit de noves amb els seus respectius comentaris.

Principals dificultats

Considerem que respecte a les dificultats podem definir tres aspectes claus. La pròpia programació, les limitacions del llenguatge i el temps real disposat per la pràctica.

En primer lloc, hem trobat dificultats amb la programació amb el 68K per múltiples raons. Era habitual a l'hora de programar haver de recercar algunes instruccions menys conegudes en el repertori i revisar les seves sintaxis i els seus funcionaments.

També influeix en les dificultats el que estem acostumats a emprar habitualment llenguatges de més alt nivell amb programes més còmodes que el simulador de 68K. A més, cal mencionar la manera en que havíem de seguir el codi mitjançant el *debugger* si havíem de trobar errors. Al començament, aquesta eina sembla estranya i complicada, encara que una vegada et familiaritzes amb ella és de gran ajuda.

En segon lloc, les limitacions del llenguatge assemblador són moltes sobre tot per les primeres idees que et venen al cap quan es pensa en voler fer un videojoc. No obstant, s'han superat aquestes dificultats.

Pel que respecta a la part de recuperació, al començament vàrem tenir alguns petits problemes. El més significant va ser que a l'hora d'afegir la finestra de hardware el programa seleccionava aquesta contínuament i no ens deixava interactuar amb el joc. Vàrem canviar el lloc on es cridava la subrutina que implementava la tasca addicional de la recuperació i així aconseguirem que la finestra de hardware s'actualitzés en segon pla i ens permetés continuar amb el videojoc.

En darrer lloc, l'aspecte més subjectiu. El temps tant d'organització com el real per poder programar el joc se'ns ha quedat curt. Les primeres opcions que vàrem barrejar no es varen poder dur a terme per diferents aspectes i finalment es va optar per aquesta opció. Som totalment conscients que si haguéssim obtingut la idea del videojoc abans haguéssim presentat un videojoc amb moltes més millores i molt més complex.

Principals afegits

En primer lloc, el principal afegit, és el canvi d'aspecte d'horitzontal a vertical el qual no és un canvi gens trivial. Doncs, el canvi a un joc vertical comporta una sèrie de modificacions per tal de controlar segons quins aspectes com pot ser el tipus de moviment dels obstacles, el lloc on es "moren" els agents d'entre altres.

La segona millora notable és l'ús de música de fons i sons tant del jugador, com de les monedes i com de les barres.

S'ha de mencionar també la part gràfica implementada, ja que al 68K tot el que es troba relacionat amb els dibuixo és realment complex i molest per programar.

Existeixen afegits com la creació de subrutines *UTIL* que ens han estat de gran ajuda però que a simple vista no s'aprecien.

En darrer lloc, mencionar la manera amb que es mostra el títol inicial i el títol de *game over* ja que la implementació de la mostra per pantalla de nombroses cadenes de text per poder generar els ASCII ARTS ens ha creat més d'un mal de cap.

Explicació de com jugar

El joc té un nivell de jugabilitat molt simple, van apareixent núvols a la part superior de la pantalla que cauen cap a la part baixa de la pantalla on es troba el jugador, que ha d'anar esquivant-les amb les fletxes esquerra o dreta per desplaçar-se a la part de la pantalla corresponent.

Els núvols tenen una longitud de mitja finestra, i apareixen a la esquerra o la dreta, igual que el jugador que únicament pot estar situat a l'esquerra o a la dreta, de forma estàtica, sense un desplaçament visible. El jugador disposa de 3 punts de vida, i cada vegada que impacta amb un núvol perd un punt, però cada vegada que el jugador aconsegueix esquivar un d'aquests, rep 5 punts a la seva puntuació.

A mida que va avançant el joc, de forma aleatòria poden caure monedes amb una velocitat més alta que la dels núvols, atorgant al jugador 5 punts si s'aconsegueix fer contacte amb una. A més d'això, cada vegada que el jugador aconsegueix esquivar 15 núvols, la velocitat a la que aquests cauen augmenta, fent cada vegada més difícil el fet de esquivar-los.

Pel que fa a la part afegida de la recuperació, algunes de les subrutines sobre el visualitzador són modificacions de la practica 1, per exemple les de mostrar la finestra de hardware, obtenir les adreces i escriure al visualitzador. Per altra part, les subrutines per obtenir el nombre de cicles en cada cas i fer el càlcul de cicles son nostres. La subrutina *getcycles* empra el TRAP 15 per obtenir el nombre de cicles, i la subrutina *countcycles* utilitza aquest nombre per obtenir la diferencia de cicles abans i després del dibuixat o de l'etapa d'actualització, obtenint així el nombre final que hem de mostrar al visualitzador.

Conclusió

En conclusió, aquesta ha estat la pràctica que més hem ens ha agradat realitzar, ja que la majoria dels treballs que hem realitzat fins ara es basen en gestió de fitxers o de objectes, i programar un videojoc era de les principals coses que teníem en ment quan començàvem a aprendre, tot i que hem tingut dificultats treballant amb aquest llenguatge, acostumats a treballar amb llenguatges més potents.

Per altre part també ens ha fet veure la facilitat que et proporciona tenir un codi ben estructurat i ordenat, ja que el codi d'alguns videojocs que vàrem cercar dins la part d'exemples de la mateixa pagina de easy68k tenien la majoria del codi dins un sol arxiu i era molt difícil de comprendre.

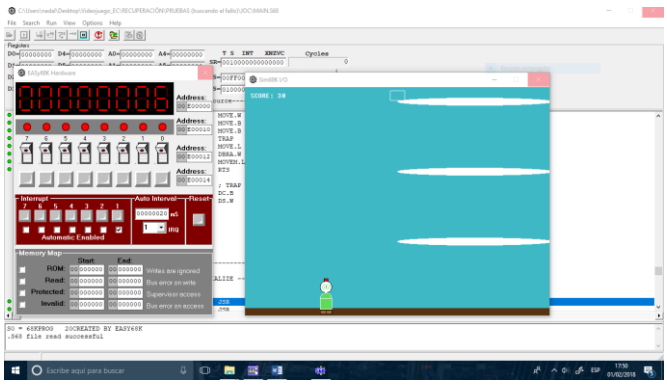
Finalment una de les coses que ens ha sorprès més d'aquesta pràctica es la capacitat del llenguatge assemblador, ja que fent el que vàrem programar a la primera part d'aquesta assignatura no pensàvem que aquest llenguatge tingués la capacitat suficient com per poder programar una cosa tan complexa com es un videojoc.
Agraïm l'oportunitat brindada en poder presentar-nos al període de recuperació i demanem disculpes pels problemes ocasionats.

Detecció del problema parlat a la tutoria

El problema que explicarem a continuació ha estat sense dubte la dificultat més gran que hem tingut. Una vegada vàrem acabar el videojoc després de realitzar algunes probes ens vàrem adonar de que el joc en segons quines situacions es "penjava" i deixava de respondre a qualsevol resposta de l'usuari i finalment es tancava el simulador.

Així doncs, intentàrem solucionar l'error i revisàrem el codi completament nombroses vegades per tal de trobar l'error, malauradament sense gaire fortuna. Decidirem demanar ajuda al professor a la tutoria i ens va explicar les situacions que ell creia que podien fallar i donar lloc a aquest problema com podien ser: la música, cadenes de caràcters no terminades en zero, trap amb una sintaxi o pas de paràmetres incorrectes etc. Hem revisat totes aquestes situacions i no hem trobat cap error.

Finalment, hem decidit intentar detectar l'error de la manera que el professor ens va explicar a la tutoria, col·locant nombres diferents a posicions específiques i veure a quin nombre s'atura quan es produeix l'error.
Primer ho vàrem fer al "Main" i els nombres ens varen indicar que el problema es trobava al STAPLOT. Tornàrem a realitzar el mateix procediment al STAPLOT i sorprenentment l'error ens va dur fins la instrucció *JSR (A0)*. Es pot observar a les imatges que es troben a continuació.



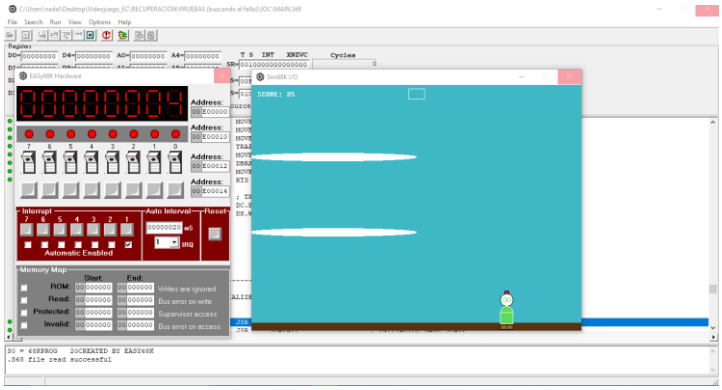
```
; ------------------------------------------------------------------------
STAPLOT
; PERFORMS STATE PLOT
; INPUT    - NONE
; OUTPUT   - NONE
; MODIFIES - NONE
; ------------------------------------------------------------------------
            MOVEM.L D0/D1/A0,-(A7)

            MOVE.L  #1,D6
            JSR     SHOWNUMBER

            CLR.L   D0

            MOVE.L  #2,D6
            JSR     SHOWNUMBER

            MOVE.W  (STACUR),D0


            MOVE.L  #3,D6
            JSR     SHOWNUMBER

            LSL.L   #2,D0

            MOVE.L  #4,D6
            JSR     SHOWNUMBER

            MOVE.L  D0,A0

            MOVE.L  #5,D6
            JSR     SHOWNUMBER

            MOVE.L  .PLTTBL(A0),A0

            MOVE.L  #6,D6
            JSR     SHOWNUMBER

            JSR     (A0)

            MOVE.L  #7,D6
            JSR     SHOWNUMBER

            MOVEM.L (A7)+,D0/D1/A0
            RTS
.PLTTBL     DC.L    STAINTRP,STAPLAYP,STAGOVRP
```

Arribats a aquest punt, deduirem tant pel comportament que hauria de tenir el videojoc com per el codi del STAPLOT, que la subrutina a la que hauria de botar en aquell instant era la de game over (STAGOVRP). Una vegada allà de manera similar a com havíem fet al STAPLOT col·locàrem les instruccions corresponents per trobar el punt de fallada.
Tornàrem a provar el joc i ens va dur fins al "punt 4" del STAGOVRP el qual es tracta d'un TRAP 15 que creiem que te la sintaxis correcta i no hauria de fallar (de fet la majoria de vegades no falla).
D'aquesta manera després de moltes proves i temps en detectar l'error concloem que es tracta de la instrucció del TRAP 15 i tal com el professor ens ha explicat a la classe de tutoria no podem fer massa més.

Adjuntem les captures de pantalla per una millor comprensió, incloem la subrutina que hem emprat per detectar el problema (SHOWNUMBER).

```
;------------------------------------------------------------------------------
SHOWNUMBER
; SHOW A NUMBER ON THE 7-SEG WINDOW
; INPUT    - THE NUMBER TO SHOW
; OUTPUT   - NONE
; MODIFIES - NONE
; ------------------------------------------------------------------------------
            MOVEM.L D0/A0-A1,-(A7)
            LEA     SEGMADDR,A0             ; GET MAPPING ADDRESSES
            JSR     GTHWADDR
            MOVE.L  (SEGMADDR), A0          ; PLACE 7-SEGMENT ADDR INTO A0
            MOVE.L  (STCHADDR), A1          ; PLACE SWITCH BUTTONS ADDR INTO A1
            MOVE.L  D6,D0
            JSR     NWRTSEGM
            MOVEM.L (A7)+,D0/A0-A1
            RTS
```

```
; ------------------------------------------------------------------------
STAGOVRP
; GAME OVER STATE PLOT
; INPUT     - NONE
; OUTPUT    - NONE
; MODIFIES - NONE
; ------------------------------------------------------------------------
            MOVEM.L A1/D0-D4,-(A7)

            MOVE.L  #1,D6
            JSR     SHOWNUMBER

            UTLSPEN #GOVSTRC
            UTLSFIL #$00000000
            MOVE.L  #2,D6
            JSR     SHOWNUMBER

            MOVE.B  #10,D3      ; NUMBER OF LINES
            LEA   .GOVSTR,A1
            CLR.W   D2

            MOVE.L  #3,D6
            JSR     SHOWNUMBER

            UTLLOCT GOVSTRX,GOVSTRY
.LOOP       ADD.W   #1,D1
            MOVE.B  #11,D0
            TRAP    #15

            MOVE.L  #4,D6
            JSR     SHOWNUMBER

            MOVE.B  #14,D0
            TRAP    #15
            ADD.W   #28,A1       ; ADD NUMBER OF CHARACTERS OF EVERY LINE

            MOVE.L  #5,D6
            JSR     SHOWNUMBER

            DBRA    D3,.LOOP
```



Demanem disculpes per les molèsties que ha suposat aquest problema i agraïm la seva comprensió.