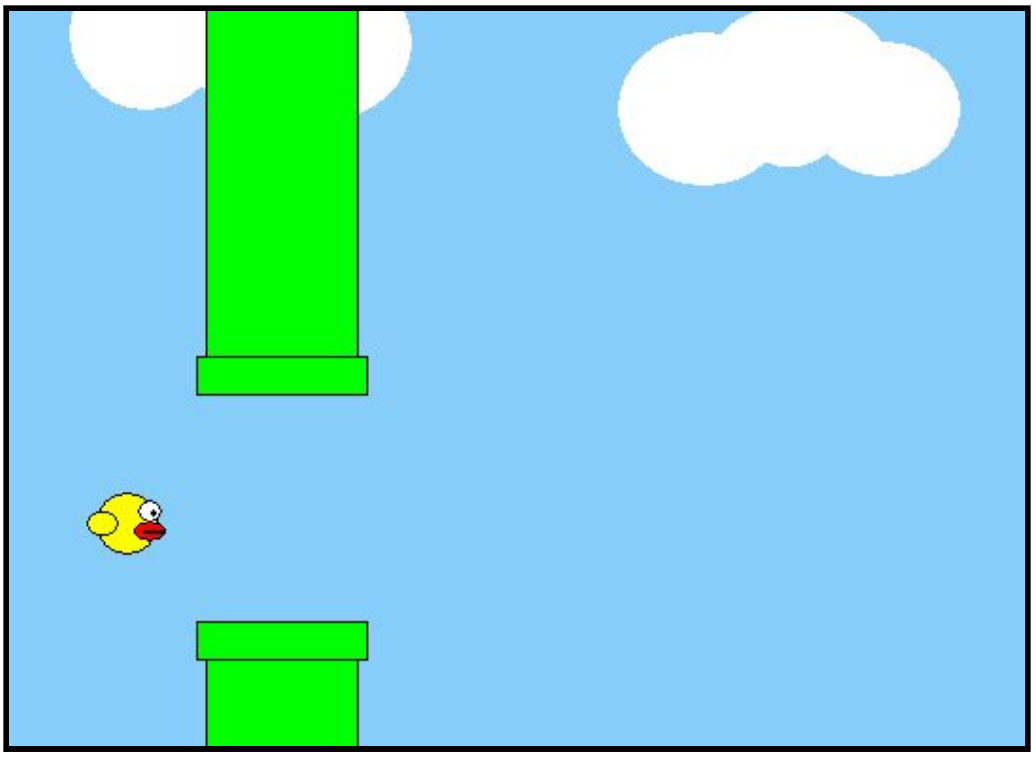
CRAZY BIRD DRIVING

THE RETURN OF THE CRAZINESS



Components del grup

- Sergi Moreno Pérez 45185320-A
- Antoni Payeras Munar 78222557-V
- Daniel Suau Vallés 45610695-Q

Introducció

En aquesta pràctica havíem de desenvolupar un videojoc o una aplicació en EASy68K emprant funcionalitats d'accés a perifèrics. El nostre grup ha decidit desenvolupar un videojoc ja que l'hem trobat més interessant i emocionat per fer.

Se'ns proporcionava una sèrie d'arxius que podíem fer servir com a base per a la implementació del joc i un conjunts de vídeos relacionats per guiar-nos per a la realització de la pràctica. També teníem accés a uns jocs amb diferents nivells de dificultat que ens servirien per prendre idees i com exemple de com s'hauria d'estructurar i implementar el nostre joc de manera adequada.

Havíem d'aconseguir implementar obligatòriament una sèrie de característiques en funció del nombre de components del grup. En el nostre cas, en ser 3 membres, de les quatres característiques que ens donaven a elegir, hem decidit implementar l'ús de fitxer i de ratolí i la visualització d'una imatge gran.

Estructura del nostre codi i afegits al codi subministrat

Per estructurar el nostre codi ens hem encarregat de crear diferentes clases per la gestió de coses diferents, per una banda hem afegit les classes MAP, PLAYER, IMGDATA, FILE, WALLS i AUDIO. Aquestes s'encarreguen de dibuixar el mapa, gestionar el moviment del jugador i el seu dibuixat; la codificació d'una imatge de gràfics o MAPBIT amb els colors dels píxels corresponents per poder posteriorment dibuixar-la; la gestió de fitxers tant per lectura com per escritura, la gestió dels murs o canonades conjuntament amb l'actualització del marcador i els seus dibuixats i finalment la implementació del audio per les fases del joc.

Al MAIN hem afegit la inicialització de l'audio i hem fet servir la variable REPAINT per només fer el plot de la imatge un sol pic a l'estat de game over.

A SYSTEM, entre les diverses inicialitzacions del sistema, també hi hem situat la inicialització del ratolí junt amb la seva corresponent ISR.

L'únic afegit a SYSCONST són la constant de trap d'audio i la constant de nivell de prioritat de interrupció de ratolí. Amb fi del seu correcte funcionament, vam haver de canviar l'ordre de prioritat entre pantalla i ratolí.

A SYSVAR, només hi hem situat les variables de ratolí que actualitzam a la ISR i que consultarem a algun punt del programa.

Als arxius CONST.X68 i VAR.X68 cal mencionar la modificació dels seus continguts per tal de tenir totes les variables i constants necessàries per al joc.

A dins la carpeta SCRIPT s'ubica el programa executable de Python, create_image.py, que emprarem per generar l'arxiu IMGDATA.X68 dins la carpeta DATA.

A la carpeta RES s'ubica la imatge a codificar que es mostra a la pantalla de mort.

A la carpeta DATA es on es genera l'arxiu IMGDATA.X68.

Principals dificultats

A l'hora de realitzar l'activitat ens surgiren molt de dubtes, entre ells el fet de com implementar que l'ocell simulés un salt pressionant la tecla de pujar. Així que cercant vàrem veure que li havíem de donar una certa velocitat inicial i a partir d'aquí, a cada cicle anar disminuint-la. Una vegada solucionat aquest problema ens va sorgir un altre, el qual consistia en com fer que l'ocell no decrementés la velocitat tant ràpidament, ja que era molt difícil jugar, així que se'ns va ocórrer el fet de mitjançant un BTST al contador de cicles, fer que disminuís la velocitat en funció del cicle en el que es trobés; no obstant això, el codi no ens acabava de convèncer. Finalment després de pensar i cercar molt, vàrem veure la solució aquesta, consistia en la implementació de la velocitat amb punt fixe com a long, per tal de destinar el primer word a les unitats de la velocitat, i el segon word destinar-lo als decimals, així podíem fer que l'acceleració fos menor. D'aquesta manera vàrem aconseguir el resultat desitjat amb una millor estructuració de codi.

Per a la visualització d'imatges ens vam adonar que el procés de mostrar una imatge de tamany 256x256 era molt lent, ja no per la optimització del nostre codi, sinó per les limitacions del processador a l'hora de dibuixar elements pixel a pixel. Degut a que per a nosaltres era ideal el tamany de 256x256, però el procés era molt lent, vam pensar en aplicar un escalat per així mostrar una imatge de tamany 128x128 en 256x256 pixels. Això ho vam aconseguir afegint operacions al bucle de dibuixat que amb el color corresponent al píxel a dibuixar, també dibuixar el contigu següent corresponent a la fila, el contigu següent corresponent a la columna i el que hi ha en diagonal a la seva esquerra i, avall, aconseguint haver dibuixat quatre píxels del mateix color. Ara només faltava fer uns canvis als bucles pel correcte funcionament d'aquests. D'aquesta manera la velocitat de dibuixat de la imatge l'hem augmentada de forma considerable sense afectar greument a la seva aparença.

Quan volíem llegir de fitxer les diferents puntuacions emmagatzemades, el que vàrem començar fent va ser guardar dins l'espai de memòria els codis ASCII necessaris per la posterior correcte visualització per pantalla. Això, però, resultava tediós i una despesa inútil de memòria. Ho vàrem solucionar afegint un bucle al plot corresponent de l'arxiu STATES, on ja havíem guardat a memòria les dades que ens interessaven, és a dir les puntuacions, i simplement anàvem visualitzant els seus bytes seguits dels caràcters desitjats fora requerir-los a memòria, com espai o salt de línia en funció del nombre de puntuacions ja mostrades a cada línea.

La visualització successiva de canonades, en primera instància, el que vam pensar fou en la creació del següent canonada en funció del cicles transcurrits. No va passar molt fins que ens vam adonar que aquesta implementació no era correcta. En el seu lloc, vàrem acabar

fent servir la posició en pantalla de la canonada predecessora, i així després de que arribés a un certa posició, podríem crear el següent.

Característiques addicionals

- Ús de fitxers

Per realitzar aquesta especificació del programa, hem decidit fer-ho a través de l'emmagatzemament a fitxer de les puntuacions, és a dir del nombre de murs superats, durant l'execució de cada partida.

- A l'arxiu FILE.X68 hi trobam 4 subrutines pel maneig dels fitxers que feim servir.
- A FILEWR, primer comprovam amb la tasca 59 si el fitxer al que volem escriure existeix o no. Això és necessari per saber si aplicar la tasca 51 o 52 per obrir el fitxer. Si existeix, feim una cridada a una altra subrutina dins aquest arxiu, FILEGET, que emmagatzemarà a la variable FILESIZE, el nombre de bytes presents al fitxer en qüestió, per així posicionar-se al final del fitxer per escriure la nova puntuació fora alterar les anteriors.
- A FILECRD, en cas de que hagi un arxiu creat amb puntuacions emmagatzemades, l'eliminarem per poder reiniciar el fitxer de puntuacions.
- A FILERD anirem llegint del fitxer de puntuacions mitjançant un bucle en el que a cada iteració llegirem dos bytes per puntuació fins al final del fitxer i les anirem guardant a una variable global. En acabar de llegir totes les puntuacions, es posa el valor del caràcter NULL al final de la string formada per facilitar la posterior visualització per pantalla.
- A FILEGET apareixerà el mateix bucle explicat al paràgraf anterior, amb la diferència de que el que emmagatzemarem a memòria és el nombre de puntuacions, de 2 bytes, que es poden llegir de fitxer.
- Finalment, a FILECAP, el que es farà és una lectura del fitxer creat amb anterioritat on es guarda una capçalera que es mostrarà per pantalla en el moment de mostrar les puntuacions.

En vistes generals del joc, l'accés a fitxers es fa cada vegada que l'ocell es xoca amb un mur, moment on escriurem la puntuació aconseguida a fitxer; i quan volguem mostrar les puntuacions aconseguides en apretar el botó de "Score" a l'estat d'intro. Tot seguit, accedim a llegir el fitxer de HEADER i el mostram per pantalla, seguit per la lectura de les puntuacions de fitxer i les mostram per pantalla.

- Visualització d'imatges

Per a implementar la visualització d'una imatge al joc primer necessitàvem la codificació amb hexadecimal de tots els píxels de la imatge en qüestió. Per fer-ho ens vàrem inspirar amb el creador de textures del joc YAR en python i el que vàrem fer fou modificar el recorregut de tota la imatge addicionant un bucle 'for' dins el que ja teníem per així recórrer no només la primera fila, sinó totes i enlloc de fer 256 iteracions, fer-ne 128, degut a que la nostra imatge es de tamany 128x128. Amb els colors un cop codificats de forma seqüencial a l'arxiu IMGDATA.X68 marcats amb la etiqueta IMGDATA, només faltava crear dos bucles, un dins de l'altre, per així, de la mateixa manera de recórrer la llista de codificacions de colors i amb ajuda del TRAP #15 i les funcions 80 i 82, anar dibuixant un a un cada píxel de la pantalla amb els respectius colors de la imatge. Aquesta part de codi referent a la visualització d'imatges ha estat afegida a l'arxiu STATES.X68 a la part corresponent a l'estat de Game Over.

- Ús de ratolí

En cas de l'ús del ratolí, hem definit a l'arxiu SYSTEM una subrutina d'inicialització de la interrupció de ratolí en la que s'habilita la IRQ pel botó up a la primera posició del vector d'interrupcions usant la constant MSEIRQ. També a aquest arxiu hem implementat la ISR del ratolí, on feim servir la variable MSECLK per indicar de l'acció de mouse i mitjançant la tasca 61 del trap 15 i amb el moviment en byte de 1 a D1, llegim l'estat de botó up i emmagatzemam la posició d'acció obtinguda a la variable MSEPOS.
Aquestes variables i constants usades les podem trobar als respectius arxius relacionats amb el sistema.
L'ús del ratolí l'hem implementat a l'estat d'introducció, on el podem fer servir per sortir del joc, per passar a l'inici del joc, per borrar totes les puntuacions emmagatzemades a fitxer o per a la visualització de puntuacions de fitxer en l'estat de puntuacions, quan també podrem fer ús de ratolí per retornar a la pantalla d'introducció. Pels casos a l'estat d'introducció, el que farem serà restaurar a 0 el valor de MSECLK i comprovarem si la posició obtinguda per la ISR i emmagatzemada a la variable MSEPOS correspon a algun dels botons existents per així executar les operacions pertinents. Si no s'hagués pitjat sobre cap botó, simplement restauraríem MSECLK. Pel cas de l'estat de puntuacions, utilitzarem el ratolí per retornar a l'estat d'intro.

Instruccions del joc

Abans de començar a jugar, s'obre una pestanya on, en cas de donar-li a al botó de PLAY, s'inicialitza el joc. Una vegada dins el joc, veurem que l'ocell, el personatge controlat pel jugador; comença a caure per acció de la gravetat, i a més se li estiran apropant unes canonades amb una velocitat constant. Es farà servir de la tecla de pujar per a aconseguir que l'ocell pugui sobreviure a les canonades passant pel buit que hi ha entre elles.
Per cada parell de canonades que superi sense xocar-se se sumarà 1 al marcador de puntuacions que apareix a la part central superior de la pantalla. Quan haguem perdut, se'ns enregistrarà la puntuació, per posteriorment poder veure-la a una taula amb totes les puntuacions aconseguides pels usuaris fins a aquest moment.

Conclusions

Per al desenvolupament d'aquest joc el primer repte fou triar el joc a desarrollar. Les limitacions tècniques del processador que l'ha d'executar, el 68K, són gaire importants ja que, a més, ho havia de fer de forma fluïda.

Aquesta pràctica ens ha ensenyat a com aprofitar totes les característiques d'aquest i exprimir-les per tal d'obtenir el resultat desitjat. Per altra banda, degut a la magnitud i complexitat del programa final, era important una bona distribució dels diferents arxius que contenen els mètodes i eines que componen la totalitat del joc. Això ens ha demostrat que de manera estructurada i clara, encara que el programa sigui molt extens, aquest es pot simplificar amb una distribució correcte del codi. A més una vegada estructurat el codi principal del joc, ara era més fàcil l'addició de noves funcions, com l'ús del ratolí, la interacció amb fitxers o la visualització d'imatges.

En definitiva, aquest projecte ha estat una forma efectiva per aprendre a aplicar els coneixements obtinguts a les classes de pràctiques junt d'altres que hem après de forma autodidacta que han estat imprescindibles per a completar la pràctica, a més d'una millora en la capacitat de cerca i resolució d'errors.