En knapp, en display, maximerad glädje - Ett Flappy Bird-projekt med Uno32

Dan Hemgren (870506-7232), Ernst Widerberg (920715-0690)

Mål och krav

Vi ska göra ett "Flappy Bird"-liknande spel i vilket en figur färdas genom en bana och måste undvika att krocka med pelare.

Spelet använder endast en knapp: medan användaren håller in denna knapp rör sig figuren uppåt, och under all övrig tid "faller" figuren nedåt. Figuren kommer att befinna sig på en konstant horisontell position och pelarna kommer att röra sig åt vänster med en konstant hastighet. Detta kommer att skapa en illusion av att figuren rör sig åt höger i en slags hinderbana.

Huvudmål

Dessa mål måste vara uppfyllda för att spelet ska ses som fungerande

- **Användaren skall kunna kontrollera spelet med hjälp av knapparna på ChipKit-enheten.**
- **När figuren krockar med en pelare skall spelet avslutas.**
- **Det ska gå att starta om efter att spelet avslutats.**

Vidare mål

Dessa funktioner kommer att implementeras i mån av tid.

- **Poängräkning**
- High score-lista
- **Start-meny**
- Teman
  - Möjlighet att ändra figurens utseende
  - Möjlighet att ändra banans utseende

Utfall

Ovanstående mål i fetstil är i skrivande stund uppfyllda. Innan projektmässan planerar vi att även implementera en high score-funktion.

Lösning

Projektet utvecklas för ChipKIT Uno32-mikrodatorn med Basic I/O-skölden. Vi kommer nyttja displayen till att visa spelet och en knapp kommer styra vår spelavatar. När knappen ej är nedtryckt sjunker spelavataren och när knappen är nedtryckt stiger avataren (motsvarande en enkel Flappy Bird-implementation). All kod kommer skrivas i C och utvecklandet kommer ske med hjälp av MCB32-verktygen. För att göra kontrollen så taktil och svarande som möjligt kommer vi nyttja interrupts för att hantera användarinput. Den inbyggda timern kommer agera klocka åt programmet sådant att spelmotorn och skärmen uppdateras i en bestämd frekvens på sextio bildrutor per sekund. I vår programloop kommer först spelmekanismer såsom kollisioner, spelar- och hinderpositioner beräknas för att sedan översättas till en 128x32-matris som sedermera används för att uppdatera skärmen. Nya hinder kommer kontinuerligt beräknas slumpmässigt (exempelvis med `rand()`) efter gränsvillkor gällande position och form sådant att spelet inte blir omöjligt men inte heller blir för lätt. Poäng representeras med ett heltalsfält och presenteras på en separat bildskräm när spelaren "dör".

Utfall

Under utvecklingsfasen kom vi fram till att en del av våra tänkta lösningar var aningen överflödiga för projektet. Det visade sig att polling av knapparna fungerade tillräckligt bra för att tillåta oss helt förbigå interrupt-hantering. Vi såg inte heller någon anledning att lägga ned arbete på att hålla programmet inom en bestämd definierad uppdateringsfrekvens utan nådde bra resultat genom en enklare sleep-metod (helt enkelt en lång for-loop).

Den stora utmaningen var att konstruera ett lätthanterligt gränssnitt för skärmen. Den 128x32-matris som vi representerade spelets tillstånd med behövdes packas byte för byte (i kolumner om åtta pixlar) över fyra rader längs skärmen. Det gränssnitt vi konstruerade gav oss ett lätt sätt att representera spelet både visuellt och spelmekaniskt med en enkel tvådimensionell matris. Eftersom spelobjekten kunde representeras i matrisen blev kollisionshantering trivial. När vi ville skriva en viss pixel till skärmen behövde vi bara kontrollera huruvida pixeln redan var aktiv. Isåfall hade något krockat.

Verifiering

Vi kommer att användartesta spelet för att bekräfta att samtliga mål under rubriken "Huvudmål" är uppfyllda. Vi kommer att testa all funktionalitet under utvecklingens gång för att hitta rätt känsla vad gäller saker som hastighet, kontrollkänslighet, etc.

Arbetsfördelning

Första steget i projektet var som sagt att skapa ett gränssnitt för att skriva till skärmen. Detta skötte Dan, med hjälp av en del kod från tidigare labbar. Resten av projektet (spellogik etc) gjorde vi tillsammans med en slags informell variant av "pair programming".

Reflektioner

Ett tidigt hinder i projektet var bristande dokumentation kring skärmens styrenhet, vilket gjorde att utvecklandet av vårt gränssnitt tog upp onödigt mycket av projekttiden. Detta gjorde att projektet upplevdes som mycket tyngre än vad det egentligen var.

När det lagret väl var på plats var det en fröjd att sätta igång med den faktiska spellogiken. Vi lärde oss mycket om C som vi inte kunde innan, och utvecklingen gick för det mesta smärtfritt framåt. Något som gjorde det här projektet roligare än många andra var att vi fick välja projekt helt själva.

Hade vi haft ännu mer tid hade vi skrivit vackrare C-kod och gärna implementerat många fler funktioner, som till exempel animation av fågeln.