Anders Mikkelsen

Innlevering 3

Oppgave:

Lage ett spill, med effektiv kode og god spillbarhet. Må bruke TFT og SD.

Avhengigheter:

- SPI
- ADXL335
- Adafruit GFX https://github.com/adafruit/Adafruit-GFX-Library
- Adafruit ST7735 https://github.com/adafruit/Adafruit-ST7735-Library

Dokumenter:

- Dette
- Fritz fil
- Fritz BB Screen Capture som .png
- Videodokumentasjon: https://www.youtube.com/watch?v=qqqWG1dX4HU
  (Filmet med mobilkamera nå, vesentlig bedre kvalitet men kutter ved 5 min så ble en litt brå slutt. Det er uansett en funksjonsvisning. Beskrivelse står nedenfor.)

Design:

Tetris! En god gammel klassiker med høy spillbarhet. Målet mitt var å følge de originale reglene i tetris og det har jeg klart! Det går kort ut på at man aldri skal kunne ha tre versjoner av en brick per 7 bricks. Man må altsa ha en pakke med 7 bricks av gangen som er en av hver type. Dette er BrickPack structen i min BrickFactory. Hver brick skal rotere rundt sin egen senter. Det gjør de også. Levels skal inkrementere for hver x antall poeng og da skal hastigheten settes opp. (Reglene for dette i denne versjonen er egenkomponert). Du får 10 poeng per plassering og 100 poeng (inkrementelt) per rad. Så 4 rader blir 100 + 200 + 300 + 400 = 1000 poeng. Hvis du passerer MAX_LEVEL som kan settes i headern til GameManager vil du vinne, hvis du fyller «tetrisbucketen» din før den tid taper du. Slår du i bordet pauser spillet. Ellers kontrollerer du brikkene med PS-controlleren.

Implementasjon:

I denne innleveringen har jeg virkelig fått jobbe med effektivisering for å få ned mengden SRAM som er i bruk samt få plass til all logikken. (Og gjøre den rask.) Dette var også intensjonen så derfor har jeg ikke fokusert på å bruke mange forskjellige komponenter slik som var intensjonen min i innlevering 2. Jeg har 44 bytes å gå på i progmem og SRAMbruk ligger nå på ca. 77% som egentlig er litt høyt og IDE'et gir advarsler men det er behørig testet og det er ingen stabilitetsproblemer selv etter mange gjentatte spillrunder så

jeg er rimelig trygg på at den holder seg. Jeg har brukt mye god tid på å minimere SRAM fotavtrykket i alle metoder. (Innenfor rimelighetens grenser).

Når jeg skulle begynne å implementere tetrismusikken innså jeg ganske fort at metoden min med timerbasert avspilling som jeg har brukt i tidligere innleveringer ikke ville fungere fordi prosesseringen av selve spillet tok litt for lang tid for de kjappeste notene. Derfor satt jeg meg inn i timers! Så jeg fikk satt opp Timer1 til å kontinuerlig oppdateres med millisekundlengden til den aktuelle noten som spilles slik at tone alltid blir resatt til riktig tidspunkt, vha. interrupts fra Timer1. Mye god læring her!

Når det gjelder effektivisering av koden for selve spillet har jeg gjort en del ting. Jeg begynte for eksempel med å lage random ider til alle bricks for hver konstruksjon av en brickpack ved å sjekke brickpacken for eksisterende ider. Dette brukte minimalt med SRAM men tok relativt sett lang tid å utføre. Derfor har jeg heller ett byte array som konstrueres for hver gang med verdien { 1, 2, 3, 4, 5, 6, 7 }. Så bare shuffler jeg det. Bruker litt mer SRAM men er vesentlig raskere. Noe mer pseudorandom men tilstrekkelig til at det funker. Grunnen til at jeg ikke har det som ett field i klassen er at jeg ønsker å frigjøre så mye SRAM som mulig til enhver tid.

I starten gjorde også updateBucketWindow() en oppdatering av hele tetrisbucketen min men dette gikk litt treigt så jeg gikk over til å lagre øverste y verdi for sist satte brick i levelmanager og kune oppdatere bucketen fra bunn og opp til det punktet for hver plassering. Ved clearing av rows setter den høyeste fylte rad til 0 slik at hele bucketen blir oppdatert. I tillegg vil metoden automatisk avslutte om den oppdager en tom rad og slik vil den «aldri» jobbe seg gjennom hele bucketen. Dette økte hastigheten betraktelig.

Jeg hadde i starten ikke color lagret som en variabel for hver bucketcontainer og det fungerte greit frem til jeg fikk rader fjernet med en ikke full rad i mellom. Da ble det problemer fordi jeg ikke bare kunne flytte staten ned. Ergo måtte jeg lagre fargen som en state i hver container. Dette krevde en del mer SRAM men det lot seg gjøre.

Den største optimaliseringen har foregått i updateBrick metoden i Screen klassen. Denne har vært igjennom mange iterasjoner! Her sjekkes først hvorvidt en gitt plass i koordinatene til en brick har verdi 1, altså skal tegnes. Hvis så tegnes den. Så justeres det for ytre venstre eller høyre om bricken fyller hele koordinatesystemet sitt i en retning. Deretter fjernes gamle tegn fra skjermen i switchen. Deretter gjøres samme kontroll for høyde som for side. Dette fører til at kun det essensielle blir tegnet. Dette har ført til veldig flytende og fin hastighet. Denne burde vært fordelt i flere metoder men jeg har ikke mer sketch plass. I tillegg kan det tenkes at det bare ville obfuskert funksjonaliteten og gjort den uleselig.

Generelt har problemene med ytelse ligget i skjermen og å unngå skriving til den så mye som mulig. Jeg har prøvd å tenke ut måter å tegne bucketen på som tegner linje for linje men problemet der er at hvert tegn har sin egen farge og dette lar seg ikke kombinere i en «string» som kan skrives over hele linjen. Så jeg tror det nåværende ytelsesnivået er så godt som det blir (Og det er mer enn godt nok synes jeg). Man kunne tenkt seg at det vil være mulig for den enkelte brick men der tror jeg kalkulering av en «string» gjør at vinningen går opp i spinningen.

Kunne tenkt meg å implementere lyder for forskjellige ting men tror det blir vanskelig når jeg bruker tone kontinuerlig for å spille musikk og timer1 til notetiming. Da måtte jeg hatt en høyttaler til og en timer til tror jeg. Ikke har jeg noe særlig sketch size igjen heller, så jeg er fornøyd! Takk for i år! Lært mye bra om effektivisering!

Fritzing:

Ingen overraskelser her. Noen av kabelfargene vil variere fra fritz til reel utgave og det er fordi enkelte av kablene mine i riktige farger er defekte så jeg tok de som fungerte. Fargevalgene på fritzen er å regne for «korrekt».

Eksterne ressurser:

Generelt: Arduino.cc

Tetris: https://en.wikipedia.org/wiki/Tetris