Breakout følgedokument

Overordnet beskrivelse av prosjekt:

Strukturen til vårt spill består av mange klasser. GameManager initialiser alle objektene og variablene som trengs i konstruktøren sin, og har en medlemsfunksjon som utfører selve game-loopen, rett etter at den starter en timer.

Game loopen

Game-loopen utfører noen enkle funksjonskall. Den tar først å leser om relevante input events har forekommet siden siste gang den sjekket, og så lenge brukeren ikke trykker på krysset på vinduet, så forsetter loopen å kjøre.

Dermed så gjøres skjermbilde i minnet om til et helt hvit bilde, game_objects.RenderObject blir kallet på som både gjør all business logikk i tillegg som at den tegner opp alle SDL_Textures som skal bli vist over til dette bildet i minne, som etterpå blir sendt til skjermen.
Til slutt så oppdaterer vi timeren, slik at vi kan holde styr på frameraten til spillet.
Vi har valgt å locke fps'en til 60 fps, for å gjøre det enklere å utvikle fysikken til spillet.

Struktur

De objektene som skal vises på skjermen arver fra en felles klasse GameObject som igjen arver fra IRenderable. Dette gjøre at disse klassene har en felles funksjon som blir kallet på i hver iterasjon av gameloopen. Vi har kombinert dette med en variant av composite-pattern for å kunne kalle på alle RenderObject funksjonene i hver iterasjon, ved kun et funksjonskall i game-loopen.

Views

Men alle objektene skal ikke alltid vises på skjermen. Vi blant annet en meny, og verken ballen, bricksene eller paddelen skal vises når menyen blir vist. Vi løste dette problemet med bruk av medlemsfunksjonspekere. Hvor vår containerklasse for GameObjects har en state, og viser forskjellige objekter basert på denne staten.

Ressurser

Vi har laget våre egne sprites til spillet. Det er ikke mange, men vi har en sprite for ballen, paddlen og bricksene. Vi benytter oss også av SDL_ttf, som lager SDL_Textures ut av skrift og en bestemt ttf-font (vi bruker minecraft fonten siden den er "retro").

Disse ressursfilene må ligge i egne directorys Sprites/ og TTF/ (working directory må være hvor alle kildefilene ligger).
Vi bruker en header-file dirent.h som er en del av libc (skrevet av GNU http://www.gnu.org/software/libc/manual/html_node/Directory-Entries.html)

Denne filen finnes ikke i normalt i visual studio. Dermed har vi funnet en fungerende dirent.h fil som fungerer for windows på nett.

Dirent.h gir oss mulighet til å scanne alle filer i et directory, som gir oss mulighet til å laste inn all ressursfilene veldig enkelt, siden vi allerede da vet navnet på dem.

Alle SDL_Textures av sprites blir lagret i en std::vector Renderer klassen. Renderer holder på en SDL_Renderer og fungerer som et grensesnitt mellom alt som skal bli vist, og denne SDL_Rendreren. Når et objekt skal bli vist på skjermen, så kaller den på en av Renderer funksjonene i sin render_object funsjon. Siden alle texturene for bildene er lagret i Renderer, så trenger kun objektene som skal vises på skjemen et offset, for å kunne velge riktig texture. Det fungerer litt annerledes med textures fra SDL_TTF, siden da blir vi nødt til å lage texture'ene i løpet av hver frame, dette er da oppgaven til FontTexture, som inneholder funksjoner som har som oppgave og kunne fleksibelt lage textures ut av strenger.

Programmeringslogikk:

GameObject er grunn-objektet som alle objekter i spiller arver i fra. Dens hensikt er å samle alle variabler og funksjoner som brukes av alle objekter. Dette inkludere også BrickContainer, som fungerer som en beholder for alle Bricks'ene i spillet. Denne ble inkludert i prosjektet etter at det grunnleggende var på plass og vi hadde fått en oversikt over hvilke variabler og funksjoner som var felles. Vi hentet her inspirasjon til denne fra Unity-kurset i 1. klasse.

Gameloopen styres av en implementasjon av Deltatime. Denne har vi satt til å skulle holde 60 fps, og ligger implementert i GameManager. Beregningen styres av SDL sin SDL_GetTicks() og sørger for at et tick på spillets game engine kun utføres når den har telt opp antall ms som skal kjøres pr. tick (1000 / 60). På siste build viste den en stabil kjøring på ~62 - 62,5 fps, og det anså vi oss som fornøyde med.

Vi begynte på en ganske massiv refaktorering av prosjektet for å få en penere og renere kode, men grunnet noen kompilerings- og rendringsfeil rakk vi ikke å få denne versjonen til før fristen. Ønsket her var at ballen skulle holde på sine verdier i form av en retning (Point) og en hastighet. GameManager tok så for seg all kommunikasjon mellom ballen og bricks/paddle, og fikk da det overordnede ansvaret for fysikk-motoren. I den versjonen flyttet vi alt som hadde med deltatime inn i Time-klassen og gameloop'en kallte så på metodene i denne klassen. Vi fikk denne versjonen til å compile, men rendring'en av objektene og interaksjonen var ikke leveringsklar. Hadde vi hatt mere tid ville en videre-utviklet utgave av blitt det endelige produktet, men vi ønsket å levere noe som fungerer 100%, og anså derfor arbeidet på denne uferdige versjonen som nyttig erfaring, da vi fikk repetert og jobbet med programvareakitektur i nok et reelt prosjekt.

Ball - Fysikk og kollisjon:

Vi låste fps, sånn at ballen og paddelen ikke beveger seg for fort, siden vi bare kan bevege den per piksel og SDL_Rect tar bare imot int og ikke double(Gir mening siden en piksel er jo en forflytting av en type int). Ballen bouncer fra veggen ved at den tar invertert x og y ved inntreffing av veggen. Dette gjør sånn at ballen har en innfalsvinkel og utgangsvinkelen der utgangsvinkel blir det invertert av innfalsvinkel. Samme gjelder bricks, bare at den kan treffe siden, toppunkt og bunnpunkt. Når det treffer paddelen så skyter den i forhold til hvor den traff på paddelen. Traff den helt til venstre av paddelen så blir vinkelen vinklet mer til vestre og samme gjelder for høyere. Hvis den treffer i midten så skyter den rett opp.

Lyd:

Vi har brukt SDL_Mixer. Det gjør sånn at vi kan bruke lyd i spillet. Treffer vegg, paddel og bricks så aktiveres lyden.

Hvordan spille Breakout:

Knappene for spillet er venstre og høyere piltast som beveger paddelen, space(SDLK_Space fungerte ikke oss meg) eller høyere ctrl for å skyte ballen ut fra paddelen, esc for å komme til menyen eller pause spillet.
Spillet fungerer sånn at ballen treffer paddelen du styrer og bouncer tilbake mot bricks du skal eliminere. Hvis du klarer å elimnere alle bricks så vinner du spillet, du taper ved å miste liv som skjer vær gang ballen går under paddelen. Menyen fungerer ved å trykke med musen på knappene.

Hva inneholder spillet:

Spillet inneholder en meny med to knapper. Start knapp og exit. Ved å trykke på start blir du tatt med til spillet som inneholder et vindu med bricks, paddel og en ball. Du har en viewport øverst som inneholder liv, frames, score og tid.

Hva er vi fornøyde med:

Alexander: Er mest fornøyd med strukturen vår, selv om den kunne vært bedre. Er utrolig fornøyd med samarbeidet og at alle er ivrig gjennom hele prosjektet. Vi har klart å dele opp alt, sånn at det blir ryddig og enkelt å legge eller forandre ting.

Gunnar: Jeg er veldig fornøyd med hva fikk gjort, i fht. hvor kort tid det er siden vi begynte å lære oss C++. Jeg hadde aldri programmert i det språket før vi begynte med det nå dette semesteret, men kombinasjonen av en spennede oppgave og erfaringen fra tidligere

semestere gjorde dette til en fornøyelig prosjekt. Jeg er dessuten veldig glad for at vi fikk til en fungerende meny, samt lyd.

Oliver: Jeg er veldig fornøyed med gameloopen, statusbaren, brikkene og paddelen. Jeg syntes det er mye lovende bak arketekturen og oppsettet.

Hva vi kunne gjort bedre:

Alexander: Vi kunne blitt ferdig med refakturering'en. Gjort ball fysikken bedre og muligens satt opp flere og bedre lyd effekter. Tatt med power-ups og tatt i bruk flere baller.

Gunnar: Jeg sier meg enig i Alexander; Jeg skulle ønske vi fikk til den siste versjonen av refakturering, men bortsett fra det er mest det kosmetiske jeg kan sette fingeren på nå på stående fot..

Oliver: Vi manglet tid og ble derfor ikke helt ferdig, derfor blir ikke alt koden brukt!

Minnelekasje

Vi brukte Visual Leak Detector for Visual C++ for å se etter minnelekkasje. Vi fikk bare opp en og det var noe i visalstudio som var problemet, den måtte loade noen filer. Skjer bare når man avslutter programmet. Viser seg å ikke være vår kode som er problemet.

Bugs

Ballen går bare igjennom paddelen, blir som et spøkelse.(Kan sikkert bare jeg som har opplevd det).
Ballen beveger seg vertikalt, bouncer ikke i veggen. Kan være pågrunn av at farten er så svak den ene retningen.
Hvis ballen er lik y aksen som paddelen men ikke x aksen, og du flytter paddelen fort nok så kan du redde deg selv.