Slutrapport

Adrian Blanco och Casper Winsnes

April 25, 2012

0.1 Programbeskrivning. Beskriv detaljerat vad programmet gör:

A:

Vårt program är en mandelbrotgenerator. Programmet kan generera bilder av mandelbrotset samt juliaset, och förhoppningsvis i framtiden andra användardefinierade funktioner. Programmet ska ha ett enkelt och minimalistiskt GUI med få knappar, i fokus ska de genererade bilderna ligga. Användare ska kunna bestämma graden av antialiasing, zoom, panorering och ha möjlighet att spara den generade bilden som en bildfil. Programmet ska dessutom ha fullskärmsfunktionalitet och fönsterfunktionalitet. Med hjälp av biblioteket Aparapi kan vi få våran javakod att konverteras till openCL och köras i grafikkortet för extra hastighet.

B: Vi lyckades göra en mandelbrotgenerator som. Det som finns på skärmen är en stor bild med ett mandelbrotset med ett fåtal knappar i hörnet som sköter olika inställningar så som fullskärm/fönster och spara bild.

De viktigaste inställningarna, antialiasingalternativ med mer, går att göra i en inställningsskärm som dyker upp då man trycker på den knapp som leder dit. I denna kan man växla mellan mandelbrotset och juliaset, vilka man måste definiera själv.

Det finns funktionalitet att köra all bildgenerering direkt på grafikkortet, dock bara om man har ett grafikkort som stöder detta.

C: Användare kan generera bilder av mandelbrotset samt definiera egna juliaset. Det finns dock inte några fördefinierade juliaset, enbart mandelbrotset.

Programmet har ett förhållandevis minimalistiskt GUI med fokus på bilderna. Alla menyknappar ligger i hörnet för att inte vara ivägen. På mac finns knapparna längst upp istället då mac implementerar knappar på ett annorlunda sätt än det vanliga.

Användare har många inställningsmöjligheter: antialiasing, detaljnivå, färg på seten, definiera egna funktioner och möjlighet att aktivera körning på grafikkortet om det finns stöd för detta på användarens dator.

Programmet har fullskärmsfunktionalitet, men det fungerar lite olika på mac/annat på grund av hur macs skärmar fungerar.

0.2 Användarbeskrivning. Vem kommer att använda ert program? Vilka antaganden gör ni om användarna? Är de vana datoranvändare, är de specialister, nybörjare, små barn?

A:

Användare ska kunna vara vem som helst, då fokus ska ligga på enkelhet i framställningen. De användargenerade bilderna med egna funktioner lär dock antagligen kräva viss kunskap om mandelbrotset för att kunna användas fullt ut. Viss kunskap om inställningsmöjligheterna (framförallt antialiasing) kan behöva förklaras för en ovanare användare.

B:

C:

0.3 Användarscenarier. Ge minst två exempel på scenarier där en av era tänkta användare använder programmet. Beskriv i detalj vad de ser, vilken typ av input de måste ge, hur de ger sin input och hur programmets output ser ut.

A:

1. Tänkt användare: tonåring/20-årsåldern med intresse för matematik. Har studerat fraktaler tidigare. Användaren vill se hur ett mandelbrotset ser ut och ha möjligheten att fokusera på de bitar som hen finner intressanta. Användaren vill även undersöka mer avancerade funktioner (exempelvis juliaset). Det första som användaren ser är det förinställda mandelbrotset som genereras vid startandet av programmet. Det enda input som krävs är musklickningar för att kunna zooma. För att spara bilder och ändra inställningar finns det självförklarande knappar att klicka på i ena hörnet, en av knapparna öppnar en meny för avancerade inställningar som sedan användaren kan välja mellan för att ändra mandelbrotsettet. För att generera mer avancerade funktioner finns några förinställda i inställningsmenyn samt en ruta för personligt input, funktionalitet för dessa är detsamma som för mandelbrotset. Programmets output är en mandelbrotfraktal som visas i bild. Bildfiler är ett valbart output om användaren vill.

2. Tänkt användare: lågstadiebarn med viss datorvana med lärare/förälder vid sidan om. Lärare/förälder vill att barnet ska få se fina bilder och mönster och står bredvid för att hjälpa till vid eventuella problem/oförståelse. Input är väldigt enkelt, varför barnet instinktivt borde klara av att hantera de grundläggande delarna av programmet bara genom att klicka sig fram. Om barnet vill ändra inställningar med mer kan det dock behöva hjälp av en äldre person som kan hjälpa det att förstå vad de olika inställningarna betyder.

B:

C:

0.4 Testplan. Beskriv hur ni tänker testa programmet. I den här uppgiften ska ni lägga extra vikt vid användartestning. Beskriv vilka uppgifter som testanvändaren ska utföra. De två användarscenarierna ska ingå i användartestningen.

A:

Scenario 1: (Tänkt användare nr. 1) Programmet ska testas genom att låta användaren får en kort beskrivning om vad programmet kan göra och överlåts sedan till att testa fritt. Några uppgifter ska genomföras av användaren: - Zooma - Ändra antialiasing - Ändra typ av bildgenerering - Spara en bild

Scenario 2: (Tänkt användare nr. 2) Programmet ska testas genom att vi visar användaren ett mandelbrotset (i programmet). Vi ber sedan användaren att gissa sig till hur man zoomar in på en punkt i programmet och ber henom även att zooma ut. Användaren ska även testa de mer avancerade funktionerna men då med viss assisstans från en mer erfaren användare av programmet.

B:

C:

0.5 Programdesign. Beskriv de grundläggande klasserna som ni avser att implementera och ge en beskrivning av de viktigaste metoderna i varje klass.

A:

MandelbrotFrame: Klassen ärver från javax.swing.JFrame och använder sig således främst av dess ärvda funktioner. Denna klass tar hand om skärmen och ska implemenetera MouseListener för att hålla koll på användarens musklick samt andra interaktioner med skärmen. Av dess metoder kommer främst MouseClicked att användas. Klassen innehåller också setFullscreen vilken gör om fönstret så att den fyller hela skärmen.

MandelbrotCanvas: Klassen ärver från java.awt.Canvas och använder sig främst av metoderna paint, zoom och render. Dessa använder sig av en BufferedImage att generera bilder på och sedan ritas ut på Canvasen. Klassen är tänkt att ritas ut på vår MandelbrotFrame. De mest använda funktionerna är not zoomIn() och zoomOut() som hanterar zoomningen.

MandelbrotGenerator: Detta är en hjälpklass som sammanfattar funktioner för GPUKernel och Antialiasingkernel eftersom dessa inte är tänkta att användaren ska interagera med. De mest använda funktionerna är nog calculate() som säger åt GPUKernel och AntialiasingKernel att arbeta, samt setAntialiasing som genom räknar ut vilka värden som ska ändras i båda kernels för att hantera ett nytt värde på antialiasing och changeSize() som används för att anpassa värdena i båda kernels till en ny fönsterstorlek.

GPUKernel: GPUKernel är en klass som ärver från com.amd.aparapi.Kernel och

genererar en mandelbrotbild som (oftast) är större än skärmen för att kunna applicera antialiasing på den. Huvudalgoritmen sitter i run() metoden, och vi använder biblioteket aparapi som konverterar all kod i den metoden till OpenCL och exekverar den i grafikkortet (eller ifall den inte kan det så exekverar den algoritmerna i processorn i multipla trådar). Eftersom vi konverterar koden till OpenCL i run() är det väldigt många restriktioner som gäller, till exempel får man inte använda objekt eller använda funktioner som finns utanför run() i de flesta fall vilket gör att det blir mycket kod och variabelduplicering, däremot är det värt det eftersom koden går mycket snabbare. Förutom run() så är några använda funktioner setMagnification() som ändrar zoomen i bilden och erase() som ritar bilden helt svart.

AntialiasingKernel: AntialiasingKernel är mer eller mindre en kopia av GPUKernel förutom själva run() metoden. run() i GPUKernel genererar en stor mandelbrotbild, medan run() i AntialiasingKernel har som funktion att applicera antialiasing genom att ta medelvärdet av färgerna i vissa områden i bilden och sedan spara dem i en mindre bild. Förutom run() så är nog den mest använda funktionen setSource som berättar för klassen vart bilden som den ska applicera antialiasing på finns.

B: MandelbrotFrame: Klassen ärver från javax.swing.JFrame och använder sig av dess ärvda funktioner. Den implementerar även java.awt.event.ActionListener samt java.awt.event.MouseListener för att känna av när användaren trycker på olika knappar och känna av var användaren klickar. Klassens främsta användning är för att skapa en skärm att visa mandelbrotseten. Eventhanteringen ser till att klassen kallar på de andra klasser som behöver känna till vad som har skett. I eventhanteringen finns även koden för att spara bilder. Det finns dessutom set- och isFullscreen() metoder som ställer in storleken på fönstret och innehållet efter att användaren valt helskärmsläge.

SettingsFrame: Klassen ärver från javax.swing.JFrame och implementerar java.awt.event.ActionListener. Klassen skapar alla knappar, sliders och rutor som behövs för användarinställningar. Då användaren trycker på ”refresh” eller ”refresh and close” uppdaterar den innehållet i huvudskärmen (en MandelbrotFrame) så att innehållet ritas ut med de nya inställningarna. Klassen är en inre klass till MandelbrotFrame för att kunna få tillgång till innehållet i MandelbrotFrame för att kunna genomföra alla nödvändiga förändringar.

MandelbrotCanvas: Klassen ärver från java.awt.Canvas och och använder sig av dess metoder för utritning. Den använder sig av en BufferedImage som den först ritar på innan denna sedan ritas ut på själva Canvasen. Det finns ett antal set/get metoder för de olika inställningana som har med utritning att göra, däribland antialiasing, overlay, savingnotification med mer. Det finns även en demometod tänkt att visa vad som går att göra med denna klass. Demometoden används inte i huvudprogrammet. zoommetoderna ligger även de här.

MandelbrotGenerator: Detta är en hjälpklass som sammanfattar funktioner för GPUKernel och Antialiasingkernel eftersom dessa inte är tänkta att användaren ska interagera

med. De mest använda funktionerna är nog calculate() som säger åt GPUKernel och AntialiasingKernel att arbeta, samt setAntialiasing som genom räknar ut vilka värden som ska ändras i båda kernels för att hantera ett nytt värde på antialiasing och changeSize() som används för att anpassa värdena i båda kernels till en ny fönsterstorlek.

OSValidator: Denna klass uppkom som en följd av att Mac OSX inte visar JComponents på samma sätt som de andra operativsystemen varför vi måste veta om programmet körs på mac eller inte. Klassen har en massa statiska metoder som returnerar true eller false baserat på datorns operativsystem.

TopRightCornerLayout: Klassen ärver från java.awt.LayoutManager och gör så att alla komponenter i en Container läggs ut på det sätt vi tänkt oss. Det finns en huvudkomponent vilken läggs ut över hela Containern och flera mindre komponenter som läggs ut från övre högra hörnet och längre mot vänster ju fler komponenter som läggs till.

GPUKernel: GPUKernel är en klass som ärver från com.amd.aparapi.Kernel och genererar en mandelbrotbild som (oftast) är större än skärmen för att kunna applicera antialiasing på den. Huvudalgoritmen sitter i run() metoden, och vi använder biblioteket aparapi som konverterar all kod i den metoden till OpenCL och exekverar den i grafikkortet (eller ifall den inte kan det så exekverar den algoritmerna i processorn i multipla trådar). Eftersom vi konverterar koden till OpenCL i run() är det väldigt många restriktioner som gäller, till exempel får man inte använda objekt eller använda funktioner som finns utanför run() i de flesta fall vilket gör att det blir mycket kod och variabelduplicering, däremot är det värt det eftersom koden går mycket snabbare. Förutom run() så är några använda funktioner setMagnification() som ändrar zoomen i bilden och erase() som ritar bilden helt svart.

AntialiasingKernel: AntialiasingKernel är mer eller mindre en kopia av GPUKernel förutom själva run() metoden. run() i GPUKernel genererar en stor mandelbrotbild, medan run() i AntialiasingKernel har som funktion att applicera antialiasing genom att ta medelvärdet av färgerna i vissa områden i bilden och sedan spara dem i en mindre bild. Förutom run() så är nog den mest använda funktionen setSource som berättar för klassen vart bilden som den ska applicera antialiasing på finns.

C:

0.6 Tekniska frågor. En lista av tekniska frågor som ni måste hantera när ni bygger ert system. Var så detaljerad som möjligt. Ett viktigt steg mot en god design är att få ner så många frågor som möjligt på papper på ett organiserat sätt med så många förslag till lösningar som möjligt.

A:

Bra algoritmer:

En stor del av vårt program bygger på matematiska algoritmer, och det krävs väldigt mycket för att få dem rätt. Framför allt så är algoritmerna som tar hand om mandelbrotsettet samt antialiasingen väldigt viktiga för programmet och därför är det mycket viktigt att inte bara få dem rätt utan att också göra dem bra eftersom de kommer iter-

eras upp emot några miljoner gånger per sekund. I många fall blir det en kompromiss mellan kvalitet och snabbhet och det är någonting man måste ta hänsyn till

GPU-acceleration:
Att få grafikkortet att räkna är inte lätt. Det finns mycket problem med trådar, väldigt anonyma crasher, restriktiva kodningsregler, och specifika hårdvarubuggar som man inte alltid kan förutse överhuvudtaget.

GUI
Trots att GUIt inte kommer vara i fokus måste det ändå vara användarvänligt och inte förvirrande, ifall våra planer på GUIt uppfyller det återstår att se.

B:

C:

0.7 Arbetsplan. Beskriv hur arbetet kommer att delas upp mellan personerna i projektet. Gör en tidsplan som visar när olika delmoment i projektet ska vara klara.

A:
Adrian arbetar med det grundläggande grafikarbetet och lågnivåkod och algoritmer/optimisering.

Casper arbetar med användarvänlighet, javaspecifika funktioner och högnivåkod samt hjälper till med grafikarbetet.

Vi har redan fått en fullt fungerande prototyp färdig, och det mesta som är kvar att implementera är användargränssnittet samt avancerade extrafunktioner.

UI är högsta prioritet i nuläget, vi måste få ett fungerande och lättanvänt gränssnitt som inte är förvirrande. Sedan efter det gäller det att koppla ihop UIt med funktionerna i MandelbrotGenerator, så att användaren själv kan ändra på bilden.

Lite mindre prioriterade funktioner är att kunna spara bilder, samt att ändra på mandelbrotalgoritmen så att den genererar ett juliaset.

Funktioner som vi har i åtanke att implementera, men inte är säkra på ifall en implementation är helt trolig är bland annat användarspecificerade funktioner, dragging av bilder.

B:

C: