Stockholms universitet

Kurs: Användning och programmering av spelmotorer, 7,5 hp

Period: VT 2015

Kurskod: SP6

Kursansvariga: Thomas Westin, Henrik Warpefelt

Inlämningsuppgifter

Allmänt

Krav

Er uppgift är att skriva en enklare spelmotor för 2D-spel - inte helt olik den som skrivs för PROG3 (Programmering i C/C++) - samt ett enklare spel som demonstrerar vad er motor kan. Dessutom ska ni läsa in er på aktuell forskning relaterad till ämnet och kritiskt granska denna.

Eftersom den här kursen är fokuserad på spelmotorprogrammering skall den däremot vara lite mer komplett än motsvarande dito för PROG3. Den måste således inkludera ett antal diskreta komponenter ni inte behövde ha tidigare. Nämligen:

- Loader
- Fysikmotor
- Ljudmotor
- Renderare
- Inmatningshantering
- Applikationsramverk

Dessa komponenter behöver inte vara av samma kaliber som motsvarigheterna i fullskaliga spelmotorer som används till AAA-spel, men det är rekommenderat att ni tittar på hur detta hänger ihop i befintliga motorer (exempelvis Unity) så ni kan få inspiration för just er implementation. Däremot är det inte meningen att ni ska använda färdiga mediabibliotek (med undantag för det som finns i Java-API:t) för att utveckla er spelmotor – ni ska alltså inte göra ännu ett wrapper-bibliotek till t.ex. SDL.

Något som förmodligen kommer upplevas som förvirrande är fysikmotorn. Den här behöver absolut inte vara avancerad utan det räcker med att man kan implementera någon form av gravitation samt kollisioner. Som ledstång kan ni ha spelet *Lunar Lander* - kan ni implementera det är er fysikmotor tillräckligt avancerad. För högre betyg bör man ha en lite mer avancerad motor - exempelvis en som kan simulera kraftöverföring.

Med applikationsramverk menar vi att motorns ska tillhandahålla klasser som implementationsprogrammeraren lätt kan ärva för att implementera sitt spel – exempelvis en ”Sprite”-klass för 2D-grafik eller en ljudklass för ljudeffekter som matas in i ljudmotorn. Dessa bör vara serialiserbara så att spelsessioner kan sparas ner till hårddisken (ni behöver inte implementera den här funktionaliteten, men förbered för den).

Er testimplementation skall skapas som ett enklare spel med en nivå som laddas in. Självklart får ni ha mer än en nivå om ni så önskar.

Spelmotorn utvecklas i två inlämningsuppgifter. Den första uppgiften går ut på att implementera de datastrukturer och ramverk som behövs för att få en fungerande spelmotor (främst applikationsramverk ovan, men det blöder över i andra delar). Den andra delen går ut på att implementera ett spel, samt de specifika subsystem som måste utvecklas för att detta skall fungera.

Den andra uppgiften bedöms *individuellt* enligt en sexgradig A-F-skala. Då det här är en inlämningsuppgift finns det ingen möjlighet till ett Fx-betyg. Se även Daisy för formella detaljer kring examinationen. En detaljerad uppgiftsbeskrivning följer i detta dokument.

Utöver detta ska ni även, i grupp och individuellt, läsa, kritisera och ge metakritik på forskning inom datorspelsfältet. Dessa uppgifter beskrivs i detalj nedan.

Krav och riktlinjer för kod

Själva koden *skall* vara skriven med Java. Vi gör tyvärr inga avsteg från det här kravet då det gör det avsevärt kan påverka hur svår uppgiften är. Vi vill även minska risken för att ni gräver ner er för mycket i programmerings-språktekniska detaljer – något som är utanför den här kursens område.

I övrigt ska följande riktlinjer följas.

- Koncisa kommentarer i koden, gärna på engelska
- Vettig och relevant namngivning av klasser, metoder, variabler, mediafiler osv

- Ingen duplicering av data, lagra inte samma data på mer än ett ställe
- Ingen duplicering av kod. Bryt om möjligt ut repeterande kod till separata metoder
- Var sparsam med räckvidden på variabler och metoder (t ex undvik global, public etc)
- Var sparsam med antalet variabler, dvs använd listor el dyl. om flera värden av samma typ ska lagras
- Håll funktionerna/metoderna enkla och korta, det gör det lättare att felsöka och återanvända funktioner
- Inga hårdkodade värden i funktionerna, dvs skicka med värden som argument istället
- Undvik om möjligt nästlade villkorsuttryck (t ex en if-sats inuti en annan if-sats)

Råd

För studenter på spelkandidatprogrammet är denna uppgift ett utmärkt tillfälle att påbörja arbetet att göra ett arbetsprov och en prototyp med spelspecifikation som kan användas i senare projektkurser.

Uppgiftsbeskrivningar

Under rubriken Allmänt anges det som är gemensamt för båda uppgifterna, samt en del övergripande information. Nedan följer således det som är specifikt för respektive uppgift.

Krav för gruppuppgiften

Gruppuppgiften består i att implementera de datastrukturer och ramverk som behövs för att få en fungerande men generell spelmotor som kan stödja alla typer av 2D-spel av motsvarande komplexitetsgrad som t.ex. *Lunar Lander*. För att få E krävs att applikationsramverket ska kunna hantera de diskreta komponenter (subsystem) som spelet kräver. Denna uppgift ger endast godkänt men ju bättre ni tillsammans lyckas skapa detta ramverk, desto bättre rustade är ni inför den individuella uppgiften.

All kod skall vara anonym. Bifoga en kortfattad projektrapport där det framgår vem som har skrivit vad. T ex klass, funktion. Projektrapporten döps till Grupp_Nr_Projektrapport.PDF, t ex ”Grupp_1_Projektrapport.PDF” och ska vara i PDF-format. 1

Utöver den praktiska delen, ska ni i grupp läsa in er på en forskningsartikel som ni letar upp via Google Scholar, eller direkt via lämpliga journaler (t ex IEEE, SIG-GRAPH etc) eller konferenser (t ex FDG, DiGRA). Observera att det ska vara publicerad forskning, inte artiklar på Gamasutra eller liknande. Om ni är osäkra bör ni fråga i Handledningsforumet i iLearn2. När ni hittat en artikel, skapar ni en tråd i Forskningsartikelforumet i iLearn2 – endast en grupp per artikel, så det är ”först till kvarn” som gäller. Därefter kan ni ta er tid att läsa artikeln och formulera en kritisk fråga eller problem med artikeln, som ni sedan skickar in

genom att svara på er egen tråd i forumet. Sammanfattningen skall vara cirka 250-500 ord lång. **Deadline 9/2 kl 10.00**.

Krav för den individuella inlämningsuppgiften

Den individuella uppgiften går ut på att expandera och specialisera motorn så att den blir speciellt lämpad för det spel som ni tänker implementera. Samt, att ge metakritik (kritik på kritiken) i forumet (mer info längst ned).

För att få E krävs det att alla 5 delarna ovan finns med och fungerar. För att använda det tidigare exemplet skall spelet *Lunar Lander* (eller dylikt) kunna implementeras med hjälp av motorn. För att kunna implementera detta måste följande saker uppfyllas:

- Spelaren skall kunna styra spelet med piltangenterna
- Objekt i världen skall kunna kollidera (exempelvis spelarens avatar och marken)
- Ljud skall spelas beroende på vad som händer i spelet (exempelvis motorljud när spelaren accelererar eller en explosion vid krasch)
- Fysik i form av gravitation som drar spelaren neråt, samt en tillfällig kraft utlöses när spelaren kör landarens motor.

Självklart måste detta i enlighet med uppgiftsbeskrivningen i stycket *Allmänt* vara segmenterat i diskreta motorkomponenter.

För högre betyg krävs en mer avancerad motor - exempelvis genom att man kan spela upp musik och ljudeffekter i flera format (midi, MP3, AAC etc), att objekt kan överföra krafter mellan varandra när dessa kolliderar eller att spelaren själv kan mappa om kontrollerna i spelet.

Man kan även höja sitt betyg genom att lägga till fler komponenter i motorn - exempelvis nätverksstöd eller skala funktionalitet för att ladda ur en bana med hjälp av *loadern* så att man kan byta bana utan att starta om spelet.

Det finns däremot ingen direkt mappning mellan en viss funktion och ett betyg. Inlämningsuppgiften kommer bedömas efter hur pass avancerad den är (i form av funktioner) och vilken kvalitet motorkomponenterna har.

OBS Koden ska även här vara anonym. På Red2 ska ni opponera på varandras arbeten; ni kommer bli tilldelade en opponent och någon att opponera på. Därför är inlämning av uppgiften en vecka innan Red2: **Deadline för kod-inlupp är den 13/3 23:59.**

För att visa att ni har tagit till er av kursinnehållet skall ni även lämna in en *individuell* rapport där ni motiverar och reflekterar över era designval utifrån kursboken, och helst också annan relevant litteratur, samt specificerar vilka delar av den utsprungliga spelmotorn som har behövt förändras och varför dessa förändringar vad nödvändiga. Rapporten bör vara cirka 3 sidor lång och förväntas i vanlig ordning att vara rättstavad och korrekturläst. Oläsliga rapporter kommer att *underkännas*, vilket medför att *den individuella uppgiften* får betyget F. Namnge rapporten som förnamn_efternamn_individuell_rapport.PDF och lämna in den i PDF-format.

Slutligen, ge metakritik (kritik på kritiken) i forumet. Ta (minst) en annan grupps sammanfattning/fråga och reflektera kring den; har de missförstått något i artikeln, något de inte tänkt på etc. Metakritiken ska omfatta 250-500 ord. Först till kvarn gäller även här, dvs ta i första hand en grupp som ännu inte fått någon kritik så att alla får feedback. **Deadline 20/3 kl 16.00.**

Råd

Följande är förmodligen självklarheter för många (eller kanske alla) som går kursen men vissa saker kan ändå vara värt att notera.

- Det absolut viktigaste att tänka på är att spelmotorer *simulerar* världen - ni ska alltså inte *emulera* världen. Det här betyder att ni mycket väl kan ”fuska” genom att förenkla fysikmotorn eller dylikt - det viktiga är att det *verkar* vara en fungerande värld.

- Gör klar programmeringen så långt som möjligt innan ni gör slutlig grafik och ljud, men experimentera med grafiken parallellt så att det inte dyker upp oväntade problem i slutet av projektet.

- Lägg inte till ny funktionalitet i spelet efter halva projekttiden. Fokusera då på att leta fel och fixa delar som inte är klara.

- Ni får självklart använda Javas inbyggda funktioner för att underlätta ert arbete - exempelvis kan det vara bra att använda Swing för att få en grund för 2D-grafiken och Javas olika filladdare för att underlätta inläsning från filer.

- Liksom att ni kan få mycket gratis genom Swing kan ni även använda Java Sound API:et (`javax.sound`) för att spela upp ljud. Man kan med lite extra arbete även få MP3 & AAC-uppspelning att fungera. Tänk dock på att ljuduppspelning kan ske något olika mellan plattformar och att ni därför kan behöva göra vissa anpassningar för att få er mod att fungera i både Windows och MacOS X.

- Genom att använda Swing får ni väldigt mycket gratis - exempelvis ett eventdrivet system för tangenttryckningar och rendering. Ni kan dessutom på ett enklare sätt inkludera menysystem och dylikt.

- Ni som läser kursen ALDA kan förmodligen få användning av vecko-uppgifterna. Försök därför reflektera över hur ni kan använda kunskaperna från den kursen i er spelmotor.

Inlämning av programmerade uppgifter

Respektive uppgift skall innan deadline (se iLearn2 för specifika tider och datum) laddas upp som ett komprimerat arkiv i ZIP- eller RAR-format till respektive inlämningsställe i iLearn2. Filerna skall heta ”Grupp Nummer Gruppuppgift.filformat” för gruppuppgiften eller ”Förnamn Efternamn individuell uppgift.filformat” för den invididuella. Exempelvis skulle detta då vara

”Grupp 0 Gruppuppgift.ZIP” respektive ”Sven B Tuba individuell uppgift.RAR”. En uppgift som lämnas in efter angivet datum kan *högst* få betyget E.

Det komprimerade arkivet skall *endast* innehålla källkod, resurser som behövs för att köra spelet, er rapport samt en README-fil som specificerar hur man startar spelet och vilka bibliotek (utöver det som följer med i J2SE) som behövs. Uppgiften kommer rättas i både Windowsmiljö (Windows 7 eller 8 64bit) och Macmiljö (OS X 64-bit). Notera att ni alltså *inte* ska skicka in era Eclipse-projekt eller dylikt. Om man måste sätta miljövariabler eller göra något mer än att packa upp och bygga med `javac *.java` *måste* detta dokumenteras i README-filen. Om byggbeskrivning saknas och kompilering enligt ovan inte fungerar kommer uppgiften att *underkännas*.

Lägg in rapporten i *rotkatalogen* i det komprimerade arkivet under filnamnet ”Rapport-Förnamn-Efternamn.pdf”. Rapporten *ska* vara i PDF-format och följa placerings- samt namngivningsstandarden – rapporter som frångår de givna instruktionerna kommer *inte* att beaktas vid rättning. Det här betyder att rapporter i exempelvis Word-, RTF-, TXT- eller PNG-format kommer att resultera i en underkänd inlämningsuppgift.