CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 07]
[Gruppmedlemmar: Nicole Nechita 20030407-7134

1. Beskrivning

Spelaren kan flytta figuren som är ett flygplan med hjälp av musrörelser. Asteroider kommer från alla sidorna och dessa måste undvikas, annars förlorar man liv. Hur mycket liv man har skrivs ner-vänster och hur mycket man har överlevt skrevs i övre-vänster. När liv når noll då stoppas varje sorts punktökning. Man kan ombörja tillstånd i spelet om man trycker var som helst på tangentbordet. När man ombörjar tillstånd liv blir full igen och tidpunkt blir 0 igen. Man kan börja tillstånd igen när som helst, även om man har inte dött än.

2. Instruktion för att bygga och testa

Man kan testa programmet som är förberedd i main genom att använda kommandot ”make”sen ” ./build/debug/play.exe” i terminalen. Just i spelet i main man kan testa kollidering, musrörelser, och tangentbord händelser. Däremot om man vill ändra på spelet i main man behöver veta vad behövs. Alla typer av objekt behöver en referens till session om dem skapas, samt en textur. Dessa absolut behövs innan objekt kan skapas. Vidare har alla objekt olika statiska default variabler som man kan ändra vid behov med funktioner, det finns funktioner som ändrar på hastighet, ljud, höjd och bredd bland andra, samt tillåta olika funktioner eller avslå vad spelet kan göra. Här behöver man matcha efter sitt behov. Detta är en spelmotor så det är ganska flexibel.

Det finns 3 rekommendationer. Efter run(), destroyTexture() för Enemy och Bullet måste anropas eftersom dem är statiska texturer och går inte att tas bort innan run()tar slut, annars förlorar alla objekt textur. Den andra rekommendation är att om man ligger på lives label, det är bäst gjort efter player objekt adderas till session. FPS går att förändra, men för stora värden funkar inte lika bra med långsamma objekt.

Alla resurser, font, audio och image hanteras från /resources i Projekt mappen med hjälp av constant::getResPath. Alla spel objekt skapas dynamisk med hjälp av getInstance().

3. Krav på den Generella Delen(Spelmotorn)

3.1. [Ja] Programmet kodas i C++ och grafikbiblioteket SDL2 används.
Kommentar: Jag har använd mig av SDL2 bibliotek och skrev detta program i C++. Inom alla filer jag har använd mig av SDL_mixer, SDL_Image, Mix_Chunk och så vidare.

3.2. [Ja] Objektorienterad programmering används, dvs. programmet är uppdelat i klasser och använder av oo-tekniker som inkapsling, arv och polymorfism.
Kommentar: Programmet har klasser, en klasshierarki och datamedlemar är privata. Det finns ett basklass och fyra klasser som ärver från det.

3.3. [Ja] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Detta stämmer. Copy och assignment är borttagna för alla objekt som kan skapas. Man kan bara skapa spelobjekt dynamisk.

3.4. [Ja] Det finns en gemensam basklass för alla figurer(rörliga objekt), och denna basklass är förberedd för att vara en rotklass i en klasshierarki.
Kommentar: Alla rörliga objekt ärver klass Component som är basklass.

3.5. [Ja] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Alla medlemmar är privata och hanters vid behöv med setter och getter.

3.6. [Ja] Det finns inte något minnesläckage, dvs. jag har testat och sett till att dynamiskt allokerat minne städas bort.
Kommentar: Detta stämmer. Alla objekt städas bort i destruktorn på rätt sätt. Objekt har textur och Session har en vector med pekare till heapen. Dessa städas bort.

3.7. [Ja] Spelmotorn kan ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.
Kommentar: Spelmotorn tar musrörelser, mushändelser och tangenten händelser för att styra spelet.

3.8. [Ja]Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.
Kommentar: Detta stämmer. Objekt kollar för kollidering löpande mot andra objekt.

3.9. [Ja] Programmet är kompilerbart och körbart på en dator under både Mac, Linux och MS Windows (alltså inga plattformspecifika konstruktioner) med SDL 2 och SDL2_ttf, SDL2_image och SDL2_mixer.
Kommentar: Har inte testat för Mac och Linux men det finns inte platformspecifika instruktioner så det borde gälla att det går.

4. Krav på den Specifika Delen(Spelet som använder sig av Spelmotorn)

4.1. [Ja] Spelet simulerar en värld som innehåller olika typer av visuella objekt. Objekten har olika beteenden och rör sig i världen och agerar på olika sätt när de möter andra objekt.
Kommentar: Det finns totalt fyra typer av spelobjekt, alla med deras egen beteende. I spelet används tre av dessa.

4.2. [Ja/] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.
Kommentar: Ett av dessa tre typer är spelare objektet som användaren kan styra med musrörelser. Den andra typ av objekt som är fienden skapas löpande hela tiden.

4.3. [Ja/] Figurerna kan röra sig över skärmen.
Kommentar: Spelarfiguren följer musen, och fiende objekt skapas i olika håll och går i olika riktingar.

4.4. [Ja] Världen (spelplanen) är tillräckligt stor för att den som spelar skall uppleva att figurerna förflyttar sig i världen.
Kommentar: Detta stämmer, spelplanet är stor nog för att spela spelet och ha roligt. Det går att få sig in i spelet och känna spännandet.

4.5. [Ja] En spelare kan styra en figur, med tangentbordet eller med musen.
Kommentar: Spelaren rör spelarfiguren med musen genom att röra på den.

4.6. [Ja] Det händer olika saker när objekten möter varandra, de påverkar varandra på något sätt.
Kommentar: Om spelare kolliderar med en fiende då fiendet dör och spelaren förlorar liv. Vid 0 liv man slutar få poäng.