CPROG Rapport för Programmeringsprojektet

[Gruppnummer: 2]
[Gruppmedlemmar: Felix Lidö – 020604]

Skriv en kortfattad instruktion för hur programmeringsprojektet skall byggas och testas, vilka krav som måste vara uppfyllda, sökvägar till resursfiler(bildfiler/ljudfiler/typsnitt mm), samt vad en spelare förväntas göra i spelet, hur figurernas rörelser kontrolleras, mm.
Om avsteg gjorts från kraven på Filstruktur, så måste också detta motiveras och beskrivas i rapporten.

Fyll i 'check-listan', så att du visar att du tagit hänsyn till respektive krav, skriv också en kort kommentar om på vilket sätt du/gruppen anser att kravet tillgodosetts, och/eller var i koden kravet uppfylls.

Den ifyllda Rapportmallen lämnas in tillsammans med Programmeringsprojektet. Spara rapporten som en PDF med namnet CPROG_RAPPORT_GRUPP_NR.pdf (där NR är gruppnumret).

1. Beskrivning

1.1. Spelmotor

Spelmotorn består av två huvud delar som båda ligger under ”springhawk” katalogen

1.2. Tillämpning

Själva sammansättning av initial komponenter ligger under ”game” katalogen och komponenterna ligger under ”scripts” katalogen

2. Instruktion för att bygga och test

2.1. Bygga

Använd make för att bygga ihop spelet. Har försökt att generalisera genom att kolla OS för att se om en kopia av resurs mappen behövs men vägen till vart både SDL och kompilator finns kan behövas ändras. Det är testat på två olika Windows maskiner och har fungerat bra på båda.

2.2. Test

Det finns två ”spel” skapade och dem kommer att spelas en efter den andra. Först är det en liten Pacman klon och sedan är ett experiment med raycasting (filerna är döpa efter wolfenstein eftersom det för själv idéen som aldrig blev klar). Du stänger ner första spelet med ESC knappen och då rensas den bort och programmet öppnar upp raycastern.

2.2.1. Pacman

Pacman styrs genom ’wasd’ och ett litet spöke (blinky) kommer att styras av sig själv. Det finns likt originalet både pellet och powerpellets att plocka upp. Vid upp plockning av powerpellet kommer spöket att ändra färg. Det finns även två röd rutor som agerar som teleportörer och fungerar som orginalet.

2.2.2. Raycaster (Wolfenstein)

Tyvärr har samma kollision check som finns i pacman spelet gjort att det troligtvis inte kommer gå att röra på sig men det går att rotera med knapparna ’e’ och ’q’

Krav på den Generella Delen(Spelmotorn)

1.1 [Ja] Programmet kodas i C++ och grafikbiblioteket SDL2 används.
Kommentar: C++ har använts under hela projektets gång tillsammans med SDL2. SDL ligger dock undangömt i motorn och om tillämpningen krävt något från SDL eller annat bibliotek ligger det i en wrapper

1.2 [Ja] Objektorienterad programmering används, dvs. programmet är uppdelat i klasser och använder av oo-tekniker som inkapsling, arv och polymorfism.
Kommentar: Data går bara att få ut genom getters/setters. Arv tillsammans med polymorfism finns under de objekt som har ”gameobject” som förälder dvs alla de objekt som kontrolleras av tillämpningsprogrammeraren.

1.3 [Delvis] Tillämpningsprogrammeraren skyddas mot att använda värdesemantik för objekt av polymorfa klasser.
Kommentar: Även om allokeringen inte skyddas så kommer programmet inte att fungera korrekt vid stack-allokering eftersom de pekare som skapas till stack-allokerade objekt kommer att hinna städas undan innan dem kommer att kunna användas.

1.4 [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: Det finns typer av basklasser. En för spelobjekt (gameobject) en för ui (uicomponent). Både ligger under ”scripts” katalogen

1.5 [Delvis] Inkapsling: datamedlemmar är privata, om inte ange skäl.
Kommentar: Basklassen gameobject ger tillgång ett par attribut som de som ärver ifrån kan ha användning av så som position, om object är en trigger eller inte och tag m.fl. Resterande klasser håller sina datamedlemmar privata

1.6 [Ja/Nej/Delvis] Det finns inte något minnesläckage, dvs. jag har testat och sett till att dynamiskt allokerat minne städas bort.
Kommentar: Allt minne som används städas bort samt att eventuella filer stängs

1.7 [Ja] Spelmotorn kan ta emot input (tangentbordshändelser, mushändelser) och reagera på dem enligt tillämpningsprogrammets önskemål, eller vidarebefordra dem till tillämpningens objekt.
Kommentar: Spelmotor har en ”input” klass som agerar som wrapper runt SDLs input system och ger ett par funktioner för att tillämpningsprogrammeraren ska kunna kolla vilka knappar som är nedtrycka och sedan agera

1.8 [Ja] Spelmotorn har stöd för kollisionsdetektering: dvs. det går att kolla om en Sprite har kolliderat med en annan Sprite.
Kommentar: Varje ”gameobject” får rätten att skriva över en funktion som heter ”onCollision” som tar in ett annat spelobjekt, likt Unity

1.9 [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: Programmet har testats att kompileras på två olika windows maskiner, men borde vara kompilerbart på Mac och Linux så länge SDL2 biblioteken hittas under kompilering processen. Det finns inga OS specfika instruktioner.

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

2 [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: Litet pacman spel där mesta från ett pacman spel. Där användaren är i kontroll av en pacman figur. Exempelvis så blir spöken ”rädda” när man plockar upp powerups samt att poäng räknaren nollställs om pacman åker in i ett spöke

2.1 [Ja] Det finns minst två olika typer av objekt, och det finns flera instanser av minst ett av dessa objekt.
Kommentar: Det finns ett par olika objekt. Det finns en spelar karaktär som kontrolleras av användaren, ett spöke som byter färg när användaren plockar upp en ”powerpellet”, samt även ”teleportPad” som ser till att om man går utanför banan så skickas man tillbaka. Man kan även baka in objekt i ”tiles” direkt i kartan som går att interagera med

2.2 [Ja] Figurerna kan röra sig över skärmen.
Kommentar: Pacman figur flyttar sig med wasd och spöket glider till höger av sig själv

2.3 [Ja] Världen (spelplanen) är tillräckligt stor för att den som spelar skall uppleva att figurerna förflyttar sig i världen.
Kommentar: Spelplanen är tillräckligt stor

2.4 [Ja] En spelare kan styra en figur, med tangentbordet eller med musen.
Kommentar: Tack vare att spelmotorn kallar på en uppdaterings funktion hos objekten och i denna funktion så kan tillämpningen använda sig av motorn input wrapper

2.5 [Ja] Det händer olika saker när objekten möter varandra, de påverkar varandra på något sätt.
Kommentar: Exempelvis så tar ”teleportPad” och förflyttar objekt till en annan position