Pacman

Mikael Larsson, Kjell Gunnarsson, Jessica Stenberg, Marcus Kristoffersson

Version 0.1

31/5/2018

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Kommentar** | **Utförare** |
| 0.1 | 24/5 2018 | Uppdaterat inledningen av SAD-dokumentet | Marcus Billgren |
| 0.2 | 28/5-2018 | Uppdaterat Arkitekturell översikt. | Jessica Stenberg |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1. Inledning**

1.1 Syfte

1.2 Målgruppen

1.3 Dokumentets omfattning

1.4 Definitioner, Akronymer och förkortningar

1.5 Referenser

**2. Arkitekturell Översikt**

2.1 Use-Case View

2.2 Use-Case Realization

2.3 Logical View

2.4 Interactions

**3. Icke-funktionella krav**

**4. Teknisk Lösning**

1. Inledning

1.1 Syfte

Det huvudsakliga syftet med projektet är att enligt vår beställares önskemål, skapa en nyversion av spelet Pacman som stöds av dagens hårdvara. Vår beställare är övertygad om att det gjordes bättre spel förr, och vill att dagens unga spelare ska ges möjligheten att få uppleva ”spel från förr - fast i ny version”.

Vår beställare hoppas att en nyversion av spelet Pacman blir en succé som kommer få spelutvecklare att börja utveckla spel utifrån att de ska vara roliga att spela och inte bara vara grafikdemonstrationer utan färdig spelbarhet.

1.2 Målgrupp

De huvudsakliga målgrupperna för detta dokument är: systemägaren(beställaren), systemarkitekterna och utvecklingsteamet

1.3 Dokumentets omfattning

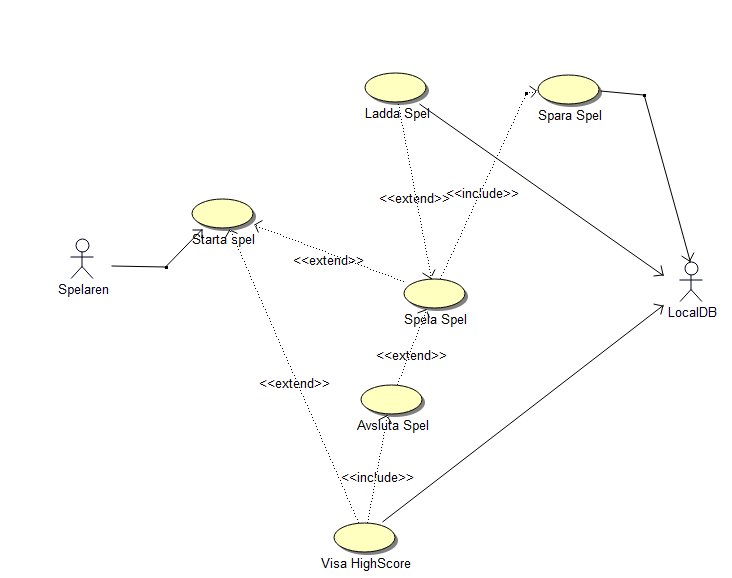
Detta dokument har för avsikt att förklara systemarkitekturen för nyversionen av Pacman samt att förklara de delar av systemdesignen som bedöms som viktiga för att utvecklingsteamet ska kunna skapa den produkt som beställaren har önskat, samt för beställaren att kunna kontrollera om de önskemål och krav som finns på produkten är inkluderade i det färdiga systemet.

1.4 Definitioner, Akronymer och förkortningar

libGDX -

1.5 Referenser

BORDE VI SKRIVA ETT PROJEKTFÖRSLAG FRÅN BESTÄLLAREN?

2. Arkitekturell översikt

2.1 Use-Case View

Bilden visar en överblick på de Use-Cases vi har i vår applikation där Spelaren är vår Actor och LocalDB är den externa aktören. Att kommunikationen mellan spelare och d0atabas fungerar felfritt är absolut prio - annars kan inga spel skapas, laddas eller sparas. Om förbindelsen inte är aktiv är menyn det enda som kommer kunna visas och systemet kommer inte kunna utföra något av valen.

Man kan även se var och vilka Use-Cases som inkluderar varandra i de olika scenariers som finns, men även de som exkluderas (inte är beroende av ett tidigare case).

2.2 Use-Case Realization

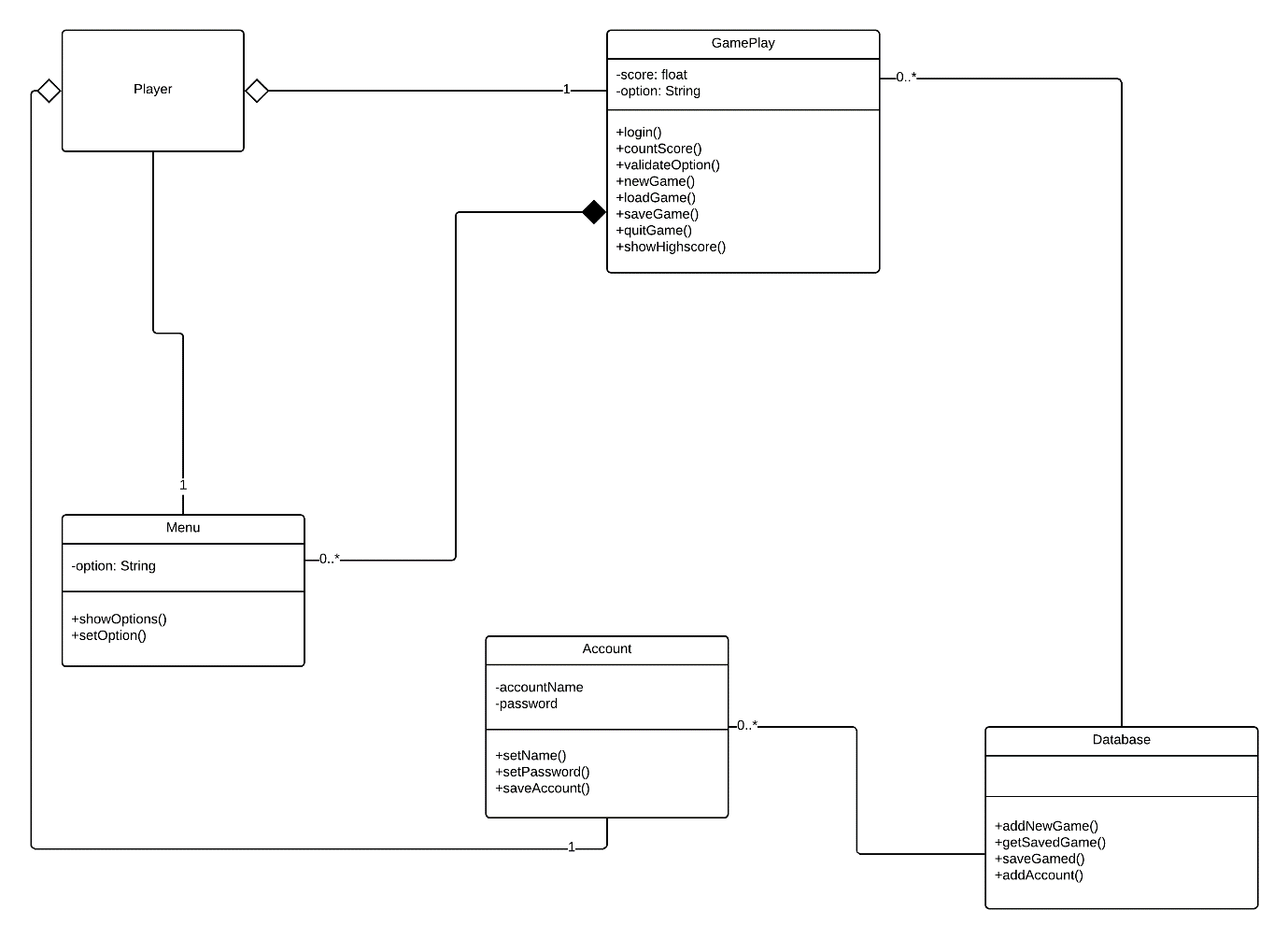
Här beskrivs några av de Use-Cases som finns tillsammans med tillhörande Story. Ett gemensamt krav som finns är bl.a. att det måste finnas tillräckligt med RAM för att spelet ska fungera korrekt.

RAMA IN USECASE – vad nu det betyder?!

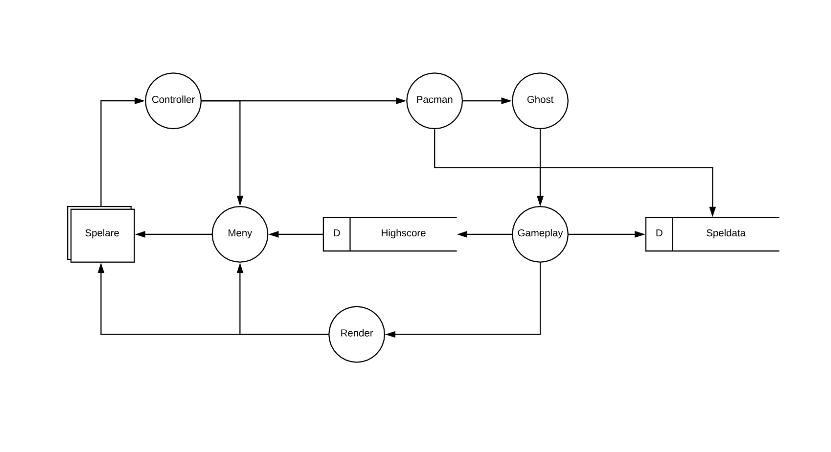
|  |
| --- |
| Use Case: Starta Spelet Actor: Spelaren Description: Användaren startar applikationen och huvudmenyn visas med alternativ att starta spelet, ladda ett befintligt spel eller titta på Highscore-listan. Story: Som användare vill jag komma till en huvudmeny där jag kan göra val när spelet startar. |
| Use Case: Ladda Spel Actor: Spelaren Description: Användaren laddar ett sparat spel från en tidigare session från huvudmenyn. Story: Som användare vill jag kunna ladda spel som jag tidigare har spelat. |
| Use Case: Spara Spel Actor: Spelaren Description: Användaren sparar alla ändringar som har skett efter den senaste sparade sessionen. Story: Som spelare vill jag kunna spara spelet för att ändringar/framsteg inte ska gå förlorade. |

2.3 Logical View

I bilden nedan illustreras några av de funktioner och klasser som bygger systemet, samt relationer dem emellan. Några klasser är beroende av andra för att de ska kunna existera, medans andra instansieras oavsett.



Nästa bild visar istället hur data skickas mellan de olika objekten/enheterna.



Översiktliga förklarningar av spelets huvudprocesser

Controller – den process som ansvarar för att ta emot input från användaren.

Pacman – Uppdaterar spelarens position på spelplanen samt kontrollerar om pacman kan äta spöken eller inte.

Meny – Den process som låter spelaren göra val att spara, ladda och avsluta när spelet är pausat eller när användaren är i huvudmenyn.

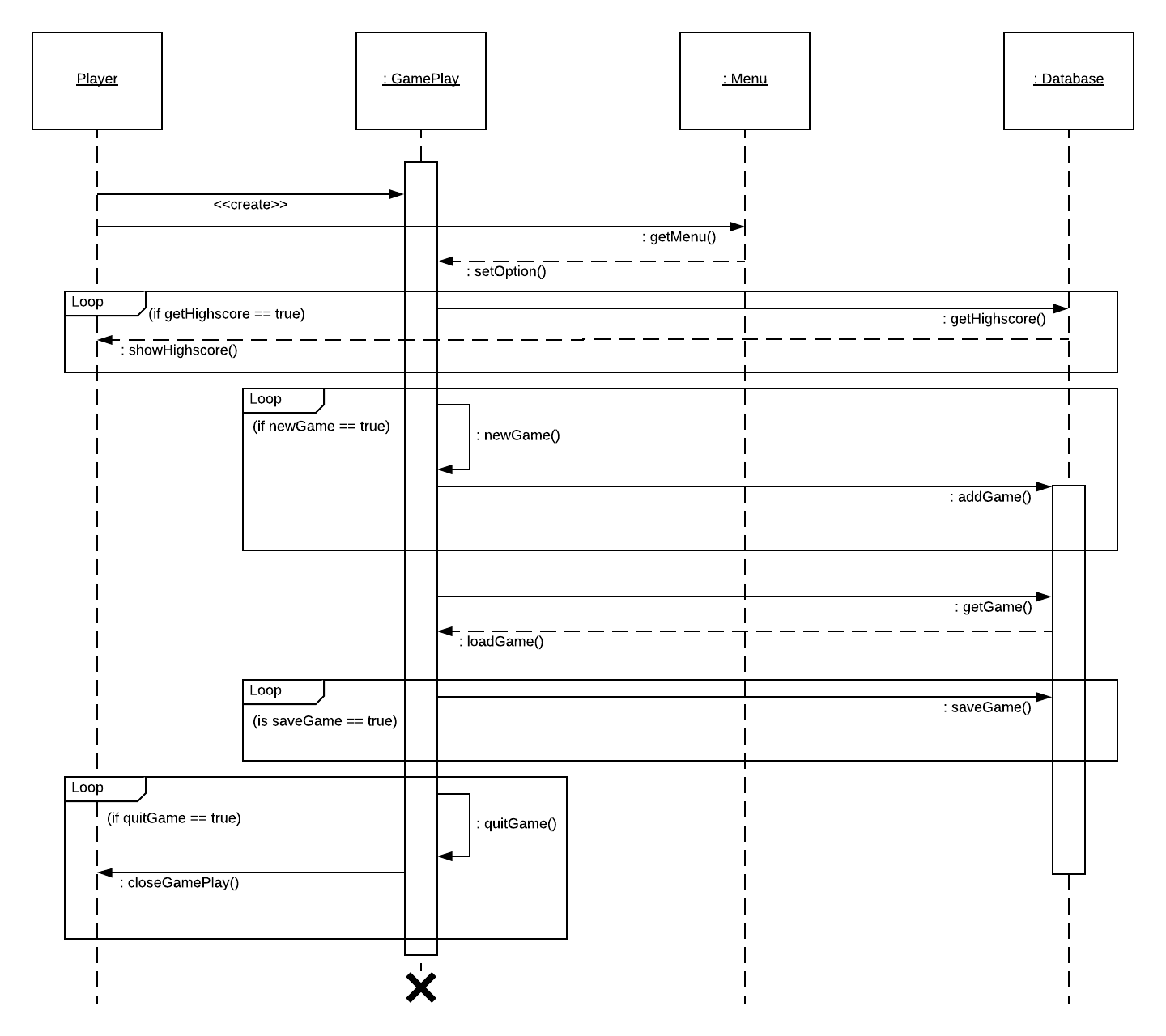
Gameplay – Den process som ansvarar för att räkna poäng, upptäcka kollisioner och ha koll på om spelaren är i livet eller inte.

Render – Renderar spelvärdens grafiska komponenter.

Ghost – Uppdaterar spökets position på spelplanen.

2.4 Interactions

Följande diagram beskriver programmet från det att spelaren har skapat ett konto eller loggat in med befintligt konto. Spelaren kan sedan göra sitt val i menyn och diagrammet visar de funktioner som körs baserat på valet.



* Efter att GamePlay har skapats så kallas funktionen getMenu() som presenterar de olika valen som finns.
* När spelaren har gjort sitt val så skickas detta in i GamePlay där valet valideras och tilldelas med setOption().
* Sedan följer olika loopar/funktioner som körs beroende av spelarens val:  
  - getHighscore() hämtar data från databasen som sedan visas för spelaren.  
  - newGame() skapar ett nytt spel som sedan läggs till i databasen.  
  - loadGame() körs när ett sparat spel har hittats och hämtats från databasen.  
  - saveGame() sparar den aktuella sessionen, skriver över tidigare version i databasen.  
  - quitGame() avslutar GamePlay och förstör även det objektet.

3. Icke-funktionella krav

|  |  |
| --- | --- |
| **IFK-** | SvarstiSystemkravder för e-tjänster |
| Krav | Förutsättning: Spelet ska kunna köras på låga systemspecifikationer samt ha stöd för operativsystem som har tillgång till JVM. Windows, Linux, MacOS, iOS, Android. |
| Realisering |  |
| Acceptanskriteria | Godkänd genomförd acceptanstest på alla operativsystem som stöds med hög prestanda. |
| Kommentar |  |

|  |  |
| --- | --- |
| **IFK-** | SvarstiInputder för e-tjänster |
| Krav | Spelet ska kunna köras med tangentbord, spelkontroller och touchscreen |
| Realisering | JXInput för XBOX 360 controller.  Java Swing, touch events för touchscreen support  [Implementera](https://docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html) en ”Key Listener” för tangentbord. |
| Acceptanskriteria |  |
| Kommentar |  |

4. Teknisk Lösning

1. Driftaspekter
   1. Databashantering:

Databashanteringen kommer att ske på användarens dator med hjälp av enkla SQLlite-databaser och därmed behövs inte en separat, fysisk server att användas. Dock så kommer det finnas krav på användarens egen hårdvara.

* 1. Programvara:

Följande programvara måste en användare ha installerad för att kunna köra spelet:

* Java Runtime 7.0 eller en senare version.
* DirectX 10.0 eller senare.
  1. Versionshantering

Projektet kommer versionshanteras via Git som är en DVCS. I varje utvecklingsmiljö finns möjligheten till att versionshantera internt. Det finns också möjlighet för versionshantering via terminalen på varje enskild enhet.

* 1. Fysisk produktionsmiljö

För att kunna köra programvarorna krävs det en dator (minimum windows 8/macOS 10.12 ”Sierra) med följande hårdvara:

* Dualcore 2.2 GHz processor.
* Ljudkort (stereo).
* Grafikkort (internt/externt) som kan köra directdraw.
* Monitor
  1. Programvaror

I vår produktionsmiljö krävs följande program:

* Libgdx (open source under Apache 2.0 licens) version 1.9.7
* Java Development Kit SE 10
* Direct X 10.0
* Android Studio (valfritt)
* Eclipse/IntelliJ som IDE för javaprogrammering.
  1. Programspråk

Följande programmeringsspråk kommer ingå i processen för att skapa spelet. Förklaringar följer varje språk.

* Java 9: Används i libgdx och är det huvudsakliga språker som kommer att användas för applikationen.
* XML: Används i Android studio för designrelaterade funktioner och är en del av spelutvecklingsmiljön.
* Kotlin: Används parallellt med java i spelutvecklingsmiljön, dock i en mindre skala och för specifika frågor.
* SQL: Används för databashantering där t.ex. användarens poäng och sparade spel placeras. MySQL används för översikt över all data.