****

**NTI Gymnasiet Kronhusgatan Göteborg**

**PacMan spel utvecklat i Java**

**PacMan game developed in Java**

av Mikael Karlgren

**Projektarbete 2015-2016**

**Tekniskt gymnasiearbete**

**Författare: Mikael Karlgren**

**Handledare: Victor Frealdsson**

**Examinator: Victor Frealdsson**

**Omfattning: Gymnasienivå**

**Datum: 2016-06-10**

Förord  
Detta är den tekniska rapporten för projektet PacMan spel utvecklat i Java. Rapporten presenterar projektets härkomst, arbetsprocess och slutresultatet. Projektet är utfärdat och utformat som tekniskt examensarbete i tredje ring på NTI Gymnasiet Kronhusgatan i Göteborg. Detta examensarbete har som syfte att presentera några av de inlärda färdigheterna från utbildningen, för att redovisa individuell kompetens om en del av utbildningsområdet.

Sveriges teknikprogram är ett brett högskoleförberedande utbildningsprogram. Alla tekniska utbildningar innefattar kunskapsutlärning rörande teknik, matematik, fysik, engelska, svenska och kemi med mera. Tekniklinjen omfattar därefter olika kurser beroende på teknikinriktning. Min inriktning, informations- och medieteknik, specialiserar sig främst på kurser rörande datorkunskap och programmering.

Jag vill tacka min programmeringslärare Igor Janco, som har lärt mig näst intill allt jag vet om Java. Utan mina Java-kunskaper lärda från hans programmeringskurser hade inte detta projekt vart möjligt.

****Sammanfattning****

Den virtuella spelindustrin är en stor och viktig exportvara för Sverige. På grund av den stora marknaden som finns är spelutveckling en väldigt attraktiv företagsamhet i dagens läge. Med hopp om att i framtiden vara en del av denna spelindustri så föddes idén. Projektet har utförts som gymnasiearbete på NTI-gymnasiet på Kronhusgatan, Göteborg.

Projektets mål var att koda ett funktionellt PacMan-liknande spel i programmeringsspråket Java. Spelet kodades från grunden på egen hand, utan att använda befintliga hjälpmedel, för att erhålla kunskaper om hur spel är uppbyggda. Syftet med projektet var att självständigt leda ett projekt och jobba med projektets alla olika fronter för att i slutändan leverera en produkt. En intressant frågeställning inför projektet var om spelet kunde kodas med kunskaper enbart lärda från Programmeringskurserna (1 och 2). Avgränsningar planerades tidigt till att inte inkludera ljud, många spelinställningar, val av svårighetsgrad, massvis med animationer med mera.

Arbetet inleddes med att inhämta information och principer från det ursprungliga PacMan-spelet. För att göra spelet unikt inhämtades idéer från spelet Agar.io. Programmeringen utfördes därefter först med hjälp av kunskaper från Programmeringskurs 1 (Java), men relativt snabbt visade det sig att informationen från denna kurs inte innehöll en passande interaktionslösning. Däremot inlärdes interaktionsförståelse (EventListeners) från Programmering 2 kursen, vilket applicerades till applikationen.

Projektet anses framgångsrikt då applikationen successivt har kodats i ren Java och fungerar riktigt bra (nerladdning [här](https://www.dropbox.com/s/o48u6wzoczsilve/PacArio.jar?dl=0), källkod [här](https://www.dropbox.com/sh/g5asobbopfg3aeo/AAD4-Si2rr8pw1hxnM4WAKHTa?dl=0)). Spelet är namngett ’PacArio’, en slags kombination av spelen **Pac**Man och Ag**ar.io.** Huvudmenyn består av tre knappalternativ vilka är ’PLAY’ / ’PLAY AGAIN’ – startar en ny spelomgång, ’GAME MODE’ – öppnar en ny meny för att välja spelläge samt ’QUIT’ – avslutar applikationen.

Arbetet som har utfärdats genom projektet har bidragit med en god insikt om hur utvecklingsprojekt går till, tillsammans med en ökad förståelse för programmering. Resultatet visar att det är möjligt att utveckla relativt komplexa program utifrån informationen som lärs ut i Programmering 1 och Programmering 2 kurserna.

PacArio, Teknik, Projekt, Programutveckling, Spelutveckling, Java, PacMan

Summary  
The virtual game industry is a big and important export market for Sweden. As a result of the huge market that exists today, virtual game developing is a very attractive business. This project’s concept was born in hope to be a part of the virtual gaming industry in the future. The project has been completed as technological secondary work assignment at NTI’s upper secondary school at Kronhusgatan, Gothenburg in Sweden.

The project’s objective was to develop a functional PacMan-like game with the programming language Java. The game was coded from scratch without using existing assistant tools, in order to earn more knowledge of game developing. The projects purpose has been to single-handedly lead a project and work with all its’ phases to deliver a finished product. An interesting question regarding development was if the game could be coded with knowledge only learnt from school programming courses (1 and 2). Limitations were early planned to not include sound, many game settings, different degrees of difficulties, tons of animations etc.

The project started with collecting information and principles from the genuine PacMan-game. To make the game more unique, ideas from the game Agar.io were collected and implemented. The coding was first written with knowledge solely from the programming 1 course (Java) but fairly soon it was clear that the course didn’t fully cover all required areas, a fitting interaction-solution was missing. This was however solved with interaction knowledge (EventListeners) learnt from the programming 2 course, which was applied to the application.

The project is considered successful, as the game step by step has been coded in pure Java, and works very well (download [here](https://www.dropbox.com/s/o48u6wzoczsilve/PacArio.jar?dl=0), source [here](https://www.dropbox.com/sh/g5asobbopfg3aeo/AAD4-Si2rr8pw1hxnM4WAKHTa?dl=0)). The game has been named ’PacArio’, as the game is a kind of combination between the games **Pac**Man and Ag**ar.io.** The mainmenu consists of 3 alternatives, ‘PLAY’ / ‘PLAY AGAIN’ – starts a new game round, ‘GAME MODE’ – opens a new menu to choose game mode and lastly ‘QUIT’ – ends the application.

The issued work throughout this project has given a good insight of how program developing projects works, and has also resulted with an increased understanding of programming. The result shows that it’s possible to develop fairly complex games with knowledge learned from the programming 1 and 2 courses.

PacArio, Engineering, Project, Software development, Game development, Java, PacMan

# INNEHÅLLSFÖRTECKNING

[BETECKNINGAR 1](#_Toc453332544)

[INLEDNING 2](#_Toc453332545)

[Bakgrund 2](#_Toc453332546)

[Mål 2](#_Toc453332547)

[Syfte 2](#_Toc453332548)

[Frågeställningar 2](#_Toc453332549)

[Avgränsningar 3](#_Toc453332550)

[TEORI 4](#_Toc453332551)

[PacMan 4](#_Toc453332552)

[Grafik 4](#_Toc453332553)

[Interaktion 4](#_Toc453332554)

[METOD 5](#_Toc453332555)

[Datainsamlingsmetoder 5](#_Toc453332556)

[Arbetsprocessen/ genomförande 5](#_Toc453332557)

[Teknikutvecklingsprocessen 6](#_Toc453332558)

[RESULTAT 7](#_Toc453332559)

[DISKUSSION 10](#_Toc453332560)

[Metoddiskussion 10](#_Toc453332561)

[Arbetsprocessen 10](#_Toc453332562)

[Designprocessen 10](#_Toc453332563)

[Resultatdiskussion 10](#_Toc453332564)

[SLUTSATSER OCH REKOMMENDATIONER 11](#_Toc453332565)

[REFERENSER 12](#_Toc453332566)

# BETECKNINGAR

Klass-metod

Metoder, det vill säga klass-metoder, är kodblock som har syftet att utfärda specifikt angivna instruktioner. Dessa kodblock innehåller olika handlingar och/eller frågor beroende på klass-metodens syfte. Genom användning av klass-metoder så behövs kodinstruktionerna endast skrivas en gång till varje funktion, sedan kan kodblocken smidigt anropas varje gång klass-metodens funktionalitet behövs.

API

API står för Application Programming Interface, vilket översätts till applikationsprogrammeringsgränssnitt. API:er är kodbibliotek som erbjuder fullt utvecklade och färdigskrivna klass-metoder, vilket programmerare kan använda i sin kod. Kodbibliotek har syftet att smidigt erbjuda klass-metoder för att förenkla och snabba på utvecklingsprocessen. Genom att återanvända redan utvecklade klass-metoder så behöver inte utvecklare koda sin egen variant för varje eftersökt funktion, utan istället sparas mycket tid genom att använda färdigkodade API metoder (Application programming interface, 2016, 21 mars).

Spellägen

Spellägen, hämtat från engelskans ”Game modes”, är ett begrepp från spelvärlden vilket innefattar en specifik spelkonfiguration av olika regler. Beroende på spellägets konfiguration så uppträder spelet på ett speciellt sätt, olikt från spelets andra spellägen.

NPC

NPC är ett spel-begrepp vilket står för Non-Player Characters, vilket översätts till icke-spelar-figurer. NPC:er är spel-karaktärer och figurer som inte är kontrollerade utav verkliga spelare, utan styrs istället av spelkoden (Non-player character, 2016, 5 april).

Lagg

Ordet ”lagg” är en teknisk term vilket innebär att en applikation har låg FPS (Frames Per Second). Om en applikation har för låg FPS (för få bilder per sekund) så upplevs applikationen ”hackig” på skärmen, det vill säga att uppdateringsbilderna förekommer i ett ojämnt flöde.

# INLEDNING

## **Bakgrund**

Den virtuella spelindustrin är en stor och viktig exportvara för Sverige. Omsättningen för spelutveckling i Sverige visades nå upp till 8,84 miljarder kronor under 2014, vilket var en ökning med 35 procent från året innan. För att framföra hur stor marknaden är så såldes det svenska spelföretaget Mojang, företaget som skapade det populära spelet Minecraft, till Microsoft för 2,5 miljarder dollar (Lundegårdh, 2015, 8 september).

På grund av den stora marknaden som finns är spelutveckling en väldigt attraktiv företagsamhet i dagens läge. Genom att få kunskap om hur spel är kodade så skapas fler möjligheter till att genomföra framtida projekt inom programutveckling.

## Mål

Projektets mål är att koda ett funktionellt PacMan-liknande spel i programmeringsspråket Java. Spelet kodas från grunden på egen hand, utan att följa ”steg-för-steg” guider eller använda kodbibliotek från tredje part. Kodbibliotek kan användas för att förenkla och snabba på utvecklingsprocessen, samt potentiellt förbättra slutprodukten, genom att erbjuda färdigkodade metoder och strukturer.

## Syfte

Arbetets syfte är att självständigt leda ett projekt och jobba med projektets alla olika fronter för att i slutändan leverera en produkt. Genom att koda ett spel från grunden så erhålls kunskap om hur spel är uppbyggda. Under arbetets gång kommer nya kunskaper om programmering och projektledning inläras. Därmed öppnas det fler möjligheter för framtida utvecklingsprojekt.

## Frågeställningar

1. Är det möjligt att koda detta spel i Java så att det körs jämnt utan lagg?
2. Kan spelet kodas med kunskaper enbart lärda från Programmering 1-kursen? Vad är skillnaden om man utvecklar spelet med kunskaper från Programmering 2-kursen i jämförelse till Programmering 1-kursen?

## Avgränsningar

Då gymnasiearbetskursen är planerad till 100 timmar och spelet mycket väl kan utvecklas inom en tidsram på över tusen arbetstimmar så krävs det begränsningar på spelets funktioner. Spelet har inget ljud och spelinställningarna är begränsade. Användaren har till exempel ingen möjlighet för att välja svårighetsgrad eller ändra upplösning på spelrutan. Spelets animationer är även begränsade i viss mån. Det finns till exempel ingen unik animation när spelets huvudkaraktär PacMan äter upp andra varelser, utan varelserna försvinner bara. Hursomhelst så är detta det enda märkbara området i spelet som saknar animationer.

Detta projekts spelversion innehåller heller inte några spökkaraktärer som i det ursprungliga PacMan-spelet. Istället är spelets NPC:er utformade som cirklar. Ytterligare så innefattar inte spelet någon typ av labyrint, utan spelplanen är uppbyggd i form av en fri rektangulär yta som endast begränsar spelaren för att röra sig utanför rektangelns kanter. Till sist så har spelets NPC:er inte någon typ av kod som omfattar komplicerad AI (Artificiell Intelligens) logik, då detta inte behövdes för spelets specifika spellogik och spellägen.

[Libgdx](https://libgdx.badlogicgames.com/index.html) är ett exempel på ett gratis kodbibliotek (API) från tredje part som är specialiserat för spelutveckling. Denna organisation erbjuder en utvecklad [API](https://libgdx.badlogicgames.com/nightlies/docs/api/) till spelutvecklare. Detta bibliotek kan användas för att underlätta arbetet i utvecklingsprocessen för detta projekt. Hursomhelst så är det bestämt att spelet utvecklas utan hjälp av tredjepartsbibliotek, för att erhålla mer programmerings-kunskap och förståelse från utvecklingsprocessen.

# TEORI

## PacMan

PacMan är ett klassiskt 2D-spel, där spelaren kontrollerar huvudkaraktären PacMan; en gul cirkel med animerad mun som ska åka runt i en labyrint och samla poäng medan spöken ska undvikas (Pac-Man, 2016, 31 mars). [www.freepacman.biz](http://www.freepacman.biz) är en hemsida där man kan spela originalspelet på nätet som demonstrerar spelets logik bra. Se bilaga 1 för en översiktlig bild på spelet.

## Grafik

Spelets grafik är indelad i en enskild klassfil ([GameWindow.java](https://www.dropbox.com/s/g8i5ds8ivc94a8y/GameWindow.java?dl=0)) vilket utvidgar sig till (extends) [JFrame](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html). Genom att inkludera JFrame till klassen så tillsätts stöd för Javas swing-arkitektur, vilket används för att skapa och hantera grafiska fönster (Oracle, Class JFrame, 2016). Från JFrame-koden startas en ny klass-fil som utvidgar sig till [JPanel](https://docs.oracle.com/javase/8/docs/api/javax/swing/JPanel.html). JPanel-klassen tillsätter ytterligare grafik-stöd till spelet genom importering av allmänna grafikkomponenter (Oracle, How to Use Panels, 2016). Från panel-klassen ritas spelets aktuella grafik.

Grafiken målas upp med hjälp av Javas grafik-metoder ([Graphics-klassen](http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html)) från Javas officiella API. Exempel på frekvent använda metoder inkluderar [#setColor](http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html#setColor-java.awt.Color-), vilket används för att välja färg att måla med och [#fillArc](http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html#fillArc-int-int-int-int-int-int-), som ritar upp och fyller i cirkelbågar (Oracle, Class Graphics, 2016). NPC grafiken (PacMans mun animation inkluderad) är gjord med #fillArc metoden, genom konstant modifiering av metodens kallelse-parametrar.

## Interaktion

Användaren styr och interagerar med spelets huvudkaraktär PacMan med hjälp av så kallade EventListeners (händelselyssnare) i spelkoden, specifikt [KeyListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/KeyListener.html) (tangentbordslyssnare), [MouseListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html) (muslyssnare) och [MouseMotionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseMotionListener.html) (musrörelselyssnare). Genom att spelapplikationen ”lyssnar” på dessa olika händelser så kan användaren kommunicera till programmet med hjälp av tangentbordet och datormusen. Handlingarna som tas emot från mus- och tangentlyssnarna översätts i spelkoden till att utföra olika funktioner beroende på inmatning och scenario.

# METOD

## Datainsamlingsmetoder

Den programmeringskunskap som har använts genom projektet härstammar huvudsakligen från Programmering 1 och Programmering 2 kurserna. Ytterligare information har hämtats online från nätet. Alla använda källor genom arbetets gång har undersökts och blivit värderade som trovärdiga källor efter noggrann inspektion.

Majoriteten av den inhämtade programmeringsinformationen utanför Programmerings-kurserna kommer från Oracles hemsida; hemsidan från det företag som utvecklar Java. Genom att hämta information direkt från utvecklaren, i detta fall Oracle, så säkerställs inlärning av korrekt information. Den inhämtade informationen från Oracle kommer mestadels från [Javas API dokumentation](http://docs.oracle.com/javase/8/docs/api/), vilket användes under utvecklingsprocessen. Under processen så har även viss information hämtats från Oracles [Java handledning sektion](https://docs.oracle.com/javase/tutorial/).

## Arbetsprocessen/ genomförande

Först så inhämtades information och principer från det ursprungliga PacMan-spelet. Informationen användes för att planera strukturen och funktioner för detta projektets alternativa version till originalspelet. När spelkonceptet var tillräckligt utvecklat så började spelet kodas i Java. Det första som kodades var spelets grundstruktur (se bilaga 3-4), vilket tidigt lades till för att underlätta utvecklingsprocessen. När spelet till slut var färdigutvecklat så kompilerades projekt-koden till en .jar fil, vilket kan köras på datorer som har Java installerat.

Under utvecklingsprocessen gjordes ytterligare konceptändringar för att göra spelet unikt. De nya principerna inhämtades mestadels från spelet [Agar.io](http://agar.io/) (se bilaga 2 för bild). Detta spel utspelas i en stor kvadratisk yta, där spelaren och alla anda karaktärer i form av cirklar åker runt på spelbanan. Om spelarens karaktär är över en annan så äter den största karaktären upp den minsta. Ändringarna med inspiration från detta spel inkluderade en fri kvadratisk navigationsyta, NPC:er i form av cirklar, och ett spelläge som heter ’Agar.io’. Detta spelläge är som Agar.io; den största figuren äter den minsta vid kontakt. Utifrån detta spelläge skapades två ytterligare spellägen för att skapa mer variation; ’Survival’ och ’Escape’, vilket byggs på samma logik med några justeringar.

## Teknikutvecklingsprocessen

Spelet kodades i Java med [Eclipses Java utvecklingsmiljö (IDE)](http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/mars2), vilket är ett datorprogram som bekvämt erbjuder viktiga programmeringsverktyg och funktioner i ett gränssnitt, vilket kraftigt förenklar programmerings-processen. För att besvara frågorna angivna i kapitel 1.4. så började spelet initialt kodas med Java kunskaper enbart lärda från Programmering 1-kursen. Med informationen från denna kurs kunde spelets bas kodas. Applikationen kodades med simpel animerad grafik som kördes jämnt och inte laggade. Efter detta steg märktes det att kunskaper endast från introduktionskursen inte räckte hela vägen.

För interaktion i spelapplikationen krävdes användning EventListeners, det vill säga händelseavlyssnare (läs kapitel 2.3), vilket lärs ut först i Programmering 2-kursen. Spelet kunde alltså inte kodas med information enbart från Programmering 1-kursen. Utan interaktionskod kunde spelaren endast styra PacMan på ett ”hårdkodat” sätt, det vill säga genom specifika anvisningar i spelkoden, och inte med mus eller tangentbord. Efter kunskap om EventListeners från Programmering 2-kursen applicerades till spelapplikationen så kan spelaren interagera med applikationen genom både mus och tangentbord.

Det bör noteras att programutvecklingen för det här projektet tog en relativt lång tid att bli färdig med; enligt projektets [loggbok](https://docs.google.com/document/d/1Mr-yvxyvnt0kAzI_joil4LE_E1Vg3WjX9ddDsQPCLf0/edit?usp=sharing) så användes 77 dokumenterade för att bli färdig med spelet. Denna tid har för det mesta använts till att förbättra slutprodukten genom att optimera spellogiken och lösa uppkomna kodproblem.

Det bör också nämnas att detta spel inte självklarts vis kan kodas av alla personer som genomgått Programmering 1 och Programmering 2 kurserna. Programutveckling skapar krav till koden att vara uppbyggd utifrån specifika strukturer och med unik logikkod, beroende på programmets funktion(er). Vad som behövs, och hur detta görs är inte självklart för alla utvecklare. I detta projekt krävde applikationen till exempel användning av en x- och y struktur för navigation, samt användning av EventListeners för fler interaktionsmöjligheter. Dessa principer planerades innan spelet började kodas, och tack vare att hjälpmedel som dessa så fungerar applikation som planerat.

# RESULTAT

Ett PacMan-liknande datorspel har utvecklats i Java, utan hjälp eller användning av kodbibliotek från tredje part. Spelet var först tänkt att vara en nerbantad variant av det originella PacMan-spelet, men under utvecklingsprocessen ändrades spelkonceptet. Detta gjorde dels för att göra slutprodukten till ett unikt spel och dels för att förenkla och förkorta utvecklingsprocessen, då projektet var planerat att genomföras på 100 timmar.

Projektet är framgångsrikt då applikationen succesivt har kodats individuellt i Java och fungerar riktigt bra (se bilaga 5-8). [Här](https://www.dropbox.com/s/07jxyqj1ew2rm3v/PacArio%20UML-diagram.png?dl=0) är ett UML-diagram som översiktligt visar applikationens kodstruktur. All kod är i princip kodad med egna förmågor utan hjälp av externa medel. Endast ett undantag har applicerats, där ett kort [kodblock](http://stackoverflow.com/questions/27706197/how-can-i-center-graphics-drawstring-in-java) inhämtades och sedan [modifierades](http://pastebin.com/hkCB0TS3). Spelet är namngett ’PacArio’, då spelets logik och utformning är en slags kombination av spelen **Pac**Man och Ag**ar.io**. Den skrivna koden kan ses och laddas ner online [här](https://www.dropbox.com/sh/g5asobbopfg3aeo/AAD4-Si2rr8pw1hxnM4WAKHTa?dl=0). Det kompilerade spelet finns uppladdat [här](https://www.dropbox.com/s/o48u6wzoczsilve/PacArio.jar?dl=0) för att laddas ner i körbart .jar format. För att öppna .jar applikationen så behöver [Java](https://www.java.com/sv/download/) vara installerat på datorn.

Applikationen är en 600 x 600 pixlar stor rektangel som är utmålad med en vit bakgrundsfärg. Det första som visas när applikationen öppnas är spelets huvudmeny (se bilaga 5). När användaren börjar interagera med applikationen genom att flytta muspekaren över spel-fönstret så börjar spelets karaktärer (cirklar) åka över skärmen, vilket passar bra som bakgrundsanimation. Huvudmenyn består av tre knappalternativ, där texten är vit i teckensnittet ’Comic Sans’ och knapparnas bakgrundsfärg är blå. När användaren håller musen över ett knappalternativ så ändras knappens bakgrundsfärg till orange, vilket indikerar att knappen är tryckbar.

**I huvudmenyn finns de tre följande alternativen;**

PLAY / PLAY AGAIN – startar en ny spelomgång med det aktuella spelläget

GAME MODE – öppnar en ny meny för att välja spelläge

QUIT – avslutar applikationen

Menyn för att välja spelläge (se bilaga 6) är likt huvudmenyn med några små ändringar. Denna meny har en titel där det överst på skärmen står GAME MODES. Knappalternativet som är det aktuellt valda spelläget har grön bakgrundsfärg, medan~~s~~ spelläge-alternativen som inte är aktuellt vald (grön) eller hovrade över (orange) har grå bakgrundsfärg. Spelläge-menyn består av fyra knappalternativ därav den sista är en tillbaka (BACK) knapp som leder tillbaka till huvudmenyn. De tre första knapparna är spelets olika spelläge-alternativ.

Alla spellägen följer samma grundprinciper. Spelaren kontrollerar en gul PacMan figur, genom att gå vänster, upp, höger eller neråt. Detta görs genom användning av piltangenterna och/eller ’W’, ’S’, ’A’ och ’D’ på tangentbordet. PacMan figurens mun har en animation som öppnas och stängs när karaktären åker runt i spelrutan. PacMan-figuren kan inte röra sig utanför applikationens kanter, men det kan spelets NPC:er. Spelomgångar början med en nedräkning, där spelaren och spelfigurerna inte kan röra sig (se bilaga 7).

Spelets form av NPC:er är kallade blobbar, vilket är cirklar i olika färger. När spelet sätter igång så åker blobbarna tvärs över spelrutan, både vertikalt och horisontellt. Blobbarnas data, det vill säga deras färg, storlek, hastighet och riktning är varje gång slumpmässigt genererad för att skapa mer variation. Den slumpmässiga data:n är delvis genererad i tillsyn till PacMans aktuella värden.

I vissa spellägen så kan PacMan äta upp blobbar. När detta händer så ökar PacMan-figuren i storlek, beroende på den uppätna blobbens storlek. Det finns en maxbegränsning på PacMans storlek vilket är 150 pixlar bred/hög. Detta är implanterat för att inte möjliggöra att spelets figurer växer sig större än applikations-fönstret. Spelet går ut på att samla ihop så många poäng som möjligt, innan spelaren förlorar. Poängtalet visas längst upp i mitten av spelskärmen. Poängtexten är 25% transparent och visas i början i en svartgrå färg, och blir röd ju mer poäng som spelaren samlar. När spelaren har uppnått 255 poäng så slutar poängtalet att bli rött, då koden är uppbyggd i RGB-format som har maxvärdet 255.

Hur blobbarna beter sig, hur spelaren samlar ihop poäng och hur spelaren förlorar är olika beroende på spelläge. **Spelets tre olika spellägen är;**

AGARIO

|  |  |
| --- | --- |
| **Antal blobbar** | 10 |
| **Vid blobb-kontakt** | Den största figuren äter upp den minsta. När en blobb blir uppäten så återuppstår en ny. |
| **Om en blobb åker över spelskärmen** | En ny blobb återuppstår. |
| **Poänginsamling** | Poäng insamlas när PacMan äter upp mindre blobbar. Poängsumman är kopplad till den uppätna blobbens storlek. |
| **Förlust** | När en blobb som är större än PacMan kommer i kontakt. |

SURVIVAL

|  |  |
| --- | --- |
| **Antal blobbar** | Spelomgångar startar med 100 blobbar. De blobbar som åker tvärs över spelrutan försvinner. |
| **Vid blobb-kontakt** | PacMan äter upp blobben och en ny blobb återuppstår. |
| **Om en blobb åker över spelskärmen** | Blobben försvinner från spelomgången och återuppstår inte. |
| **Poänginsamling** | Poäng insamlas när PacMan äter upp blobbar. Poängsumman är kopplad till den uppätna blobbens storlek. |
| **Förlust** | När alla 100 blobbar har åkt över spelrutan och det inte finns några kvar. |

ESCAPE

|  |  |
| --- | --- |
| **Antal blobbar** | Spelomgångar startar med 15 blobbar. Fler blobbar läggs till desto längre spelaren överlever (+1 blobb varje 3,2 sekund). |
| **Vid blobb-kontakt** | PacMan blir uppäten och spelaren förlorar. |
| **Om en blobb åker över spelskärmen** | En ny blobb återuppstår. |
| **Poänginsamling** | Poäng insamlas medans PacMan lever (+1 poäng varje 0,2 sekund). |
| **Förlust** | När PacMan kommer i kontakt med en blobb. |

När spelaren förlorar så dör PacMan, och spelaren kan inte längre styra den aktuella PacMan-figuren. När detta sker så ändras PacMan ät-animation till att helt äta upp sig själv. Efter att PacMan har ätit upp sig själv så visas titeln Game Over över huvudmenyn, och texten på menyns första knappalternativ ändras från PLAY till PLAY AGAIN (se bilaga 8).

# DISKUSSION

## Metoddiskussion

All inhämtad information kom från trovärdiga källor, vilket resulterade till korrekt fakta. Då majoriteten av källorna hämtades från nätet så erhölls information smidigt och snabbt, och flera faktakällor kunde jämföras för att säkerställa genuin information. Det var positivt att spelets grundfunktioner var bestämda innan utvecklingsprocessen började i avsikt till att planlägga ett koncept för slutproduktens resultat.

## Arbetsprocessen

Arbetstiden har främst använts för utvecklingsprocessen, det vill säga för Java-programmeringen. Detta programmeringsspråk har använts i tidigare projekt och därmed förenklades arbetsprocessen, då viss kod kunde återanvändas och potentiella kodproblem som nybörjare skulle fastna med förhindrades. Då applikationens kodstruktur tidigt kodades så framskred utvecklingen smidigt.

## Designprocessen

Projektets design finns både som grafiskt gränssnitt och speldesignsuppbyggnad. Applikationens grafiska gränssnitt är simpelt med enkla former men väluttänkt. Principer inhämtade från gränssnittsdesign-kursen har applicerats i samband till applikationens grafiska utformning. Speldesignen uppkom från en individuellt skapad kombination av speluppläggen från PacMan och Agar.io spelen, vilket resulterade i ett unikt koncept.

## Resultatdiskussion

Spelets slutprodukt kunde inte blivit mycket bättre med förutsättningarna som fanns. Vissa funktioner som planerades initialt implanterades dock inte till applikationen, då dessa uteblivande funktioner var väldigt tidskrävande. Med facit i hand så borde spelplanen ha varit bättre uttänkt, då flera speländringar förekom under utvecklingsprocessen.

De områden som skulle kunna förbättras i det aktuella spelets utformning är spelanimationerna, då dessa inte alltid finns tillgängliga för alla lägliga spelhandlingar, till exempel när PacMan-figuren äter upp blobbar. Dessutom kunde spelets belöningssystem varit bättre utformat genom att inkludera ett riktigt vinst-scenario, då spelet inte kan vinnas och spelarens enda motivation är att samla ihop mer poäng.

# SLUTSATSER OCH REKOMMENDATIONER

Projektets resultat är en applikation i form av ett spel ([nerladdning här](https://www.dropbox.com/s/o48u6wzoczsilve/PacArio.jar?dl=0)), vilket främst efterliknar det originella PacMan-spelet. Viss inspiration har även inhämtats från spelet Agar.io. Spelet har framgångsrikt kodats i programmeringsspråket Java, och fungerar dessutom väldigt bra i praktik vilket ses när man kör spelet. Resultatet av arbetet har bidragit med en god insikt om hur utvecklingsprojekt går till med alla faser, från idé till slutprodukt. Utvecklingsprocessen har även bidragit till en ökad förståelse för programmering i allmänhet. Resultatet visar att det är möjligt att utveckla relativt komplexa program, som detta projekts specifika spelkoncept, utifrån informationen som lärs ut i Programmering 1 och Programmering 2 kurserna.

Koden bygger huvudsakligen på innehållet från Programmering 1 kursen. Med informationen från denna kurs kunde applikationsgrafiken framgångsrikt animeras i koden utan lagg, det vill säga med ett jämnt flöde av bilduppdateringar. Applikationen kunde dock inte fullt ut kodas enbart med kunskaper från Programmering 1 kursen, på grund av brist för en passande interaktionslösning. Utan detta var spelarens karaktärnavigation kraftigt begränsad, då spelarens karaktär endast kunde styras statiskt genom hårdkodade anvisningar i spelkoden. Från Programmering 2-kursen inlärdes information om händelseavlyssnare (EventListeners), vilket används för att lyssna på element så som mus- och tangentbordhandlingar. Dessa medel applicerades till applikationen, vilket ledde till att användaren istället dynamiskt kan interagera med applikationen genom användning av tangentbord och datormus.

Vissa element kunde ha lagts till för att förbättra spelets användarupplevelse, till exempel fler animationer och ljudstöd. Implantering av dessa funktioner undveks, även om implantering av ljud är relativt enkelt, då utvecklingsprocessen redan hade tagit upp mer tid än planerat. Projektets upplägg borde vart bättre planerat och förståelse angående den långa process som programutveckling ofta kräver, som vid detta fall, borde ha tänkts på vid planeringen.

För utvecklingsprojekt bör programmets funktioner, utformning och arbetets upplägg noggrant planeras innan programutvecklingen startar. För att försnabba utvecklingsprocessen så är det även rekommenderat att återanvända kod, använda redan utvecklade kodbibliotek, samt att ha tillgång till att prata och få hjälp från kunniga personer ifall man åker fast; speciellt vid kommersiella syften där tid spelar stor roll.

# REFERENSER

Application programming interface. (2016, 21 mars). I *Wikipedia*. Hämtad 2016-04-03,

<https://en.wikipedia.org/wiki/Application_programming_interface>

Lundegårdh, P. (2015, 8 september). Nu säljer Sverige mer spel än el. *Di Digital*.

Hämtad 2016-05-08, från <http://digital.di.se/artikel/nu-saljer-sverige-mer-spel-an-el>

Non-player character. (2016, 5 april). I *Wikipedia*. Hämtad 2016-04-03,

<https://en.wikipedia.org/wiki/Application_programming_interface>

Oracle. (2016). *Class Graphics*. Hämtad 2016-04-03, från

<http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html>

Oracle. (2016). *Class JFrame*. Hämtad 2016-04-03, från

<https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html>

Oracle. (2016). *How to Use Panels*. Hämtad 2016-04-03, från

<https://docs.oracle.com/javase/tutorial/uiswing/components/panel.html>

Pac-Man. (2016, 31 mars). I *Wikipedia*. Hämtad 2016-04-03, från

<https://en.wikipedia.org/wiki/Pac-Man>

# BILAGOR

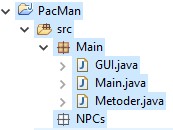
Bilaga. 1



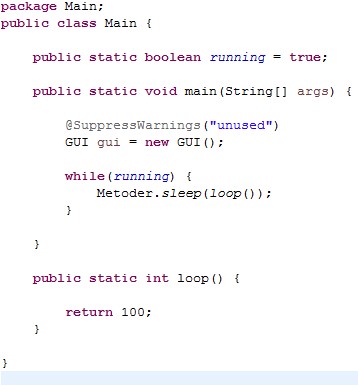
Bilaga. 2



Bilaga. 3



Bilaga. 4



Bilaga. 5



Bilaga. 6



Bilaga. 7



Bilaga. 8

