Projektbeskrivning

Katt och mus

2025-06-11

Projektmedlemmar:
Victoria Bergendahl vicbe988@student.liu.se

Handledare:
Morgan Nordberg < morno368@student.liu.se>

Innehåll

1. Introduktion till projektet....2
2. Ytterligare bakgrundsinformation....2
3. Milstolpar....2
4. Övriga implementationsförberedelser....4
5. Utveckling och samarbete....4
6. Implementationsbeskrivning....6
   6.1. Milstolpar....6
   6.2. Dokumentation för programstruktur, med UML-diagram....6
7. Användarmanual....7

Projektplan

1. Introduktion till projektet

Jag utvecklar ett 2D-spel där du som spelare styr en karaktär, i detta fall en katt, genom en värld uppbyggd av olika miljöer så som gräs, vatten, trä och sten. Spelet är skapat i programmeringsspråket Java och är designat för att vara lätt att komma in i, även om du inte har spelat mycket tidigare. Tanken är att man ska få en rolig upplevelse som passar bra som en kort paus eller för dig som gillar 2D-spel.

Spelet går ut på att ta sig genom en värld uppbyggd av tiles, vilket är små fyrkantiga ”byggstenar” som formar kartan, för att fånga en mus som försöker hålla sig undan. Vissa tiles, som gräs, är fria ytor där spelaren kan röra sig hur den vill, medan andra, så som vatten, trä och sten, fungerar som hinder som blockerar vägen. Du styr katten med piltangenterna w, a, s, d och utforskar kartan för att fånga musen. Men musen är smart och håller sig undan från dig, vilket gör jakten till en liten utmaning.

Spelet har ett tydligt mål vilket gör det lätt att förstå. Katten kan inte gå genom hinder (trä, sten eller vatten), tack vare en funktion som kallas kollisionshantering. Om du lyckas fånga musen vinner du spelet, och då kommer testen "You Won!" upp på skärmen. Det handlar alltså om att navigera kartan, planera dina steg och tajma dina rörelser för att överlista musen.

2. Ytterligare bakgrundsinformation

Här är en sammanfattning av den viktigaste informationen som krävs för att förstå spelet:

Spelets värld är konstruerad med hjälp av tiles, små fyrkantiga bilder som representerar olika typer av terräng, såsom gräs, vatten, trä och sten. Varje tile har specifika egenskaper, till exempel om den är genomskinlig eller blockerar rörelse (kollision). Denna struktur gör det möjligt att skapa en dynamisk och varierad spelvärld där olika delar av kartan kan påverka hur spelaren och andra objekt rör sig.

För att säkerställa att spelaren, en katt, inte kan gå igenom hinder som väggar eller vatten, används kollisionsdetektering. Systemet kontrollerar om kattens nästa position överlappar med en tile som är markerad som ett hinder. Om så är fallet, stoppas rörelsen i den riktningen. Detta är en grundläggande mekanik som gör spelvärlden realistisk och interaktiv.

Spelaren (katten) och musen är objekt som har position, hastighet och riktning. Katten styrs manuellt med tangentbordet, medan musen rör sig automatiskt och försöker hålla avstånd från katten genom att fatta beslut baserat på deras relativa positioner. Denna logik är central för spelets dynamik och utmaning.

Spelet använder Java-biblioteket Graphics2D för att rita upp spelvärlden, spelaren och musen på skärmen. Rendering görs effektivt genom att bara visa det som ligger inom skärmens synfält, baserat på spelarens position. Detta optimerar prestandan och ger en smidig visuell upplevelse.

Spelet drivs av en loop som kontinuerligt uppdaterar positioner, hanterar kollisioner och kontrollerar om spelaren har vunnit (t.ex. genom att fånga musen). Denna loop körs i en separat tråd för att hålla spelet responsivt och undvika fördröjningar.

3. Milstolpar

#	Beskrivning
1	Preliminärt bestämma hur spelat ska bli.
2	Få upp ett fönster på skärmen.
3	Annan färg på fönstret.
4	Visa en ”spelare” på skärmen.

5	Få spelaren att röra på sig med hjälp av tangenter.
6	Rita katt och bakgrund tiles.
7	Implementera en bild (katt) på spelaren.
8	Implementera bakgrund.
9	Implementera så att katten alltid är i mitten av skärmen.
10	Implementera kollisionshantering mellan tiles och katt.
11	Rita en mus.
12	Implementera musen på mappen.
13	Implementera kollisionshantering mellan katt och mus.
14	När musen är fångad ska spelaren inte kunna röra på sig längre.
15	När spelaren fångat musen ska det komma upp en ruta som visar att man har vunnit.
16	Få musen att röra sig från spelaren.
17	Gör en ordentlig bana (stor)
18	Göra banor med olika svårighetsgrad
19	Göra så att spelaren kan bli snabbare genom att plocka upp något
20	
21	
22	
23	
…	

4. Övriga implementationsförberedelser

Klasser jag kommer vela implementera:

- Entity (abstract): Funktion för alla karaktärer.
- Player: Själva spelarens klass med info om hur den fungerar.
- Mouse: Klass för musen och hur den fungerar.
- Tile: För att visa en bild och tilens kollision.

- TileManager: För att hämta ”tilesen” och visa kartan.
- CollisionControll: Hantera kollision.
- GamePanel: För att uppdatera hur bakgrund och karaktär rör sig.
- KeyHandler: För att kunna röra sin spelare med hjälp av w,a,s & d.
- LaunchPage: För att visa själva rutan.
- Direction (enum): Innehåller upp, ner, höger och vänster.
- IGameState (interface): Interface för när spelaren vunnit (hasWon).
- GameState: Klass för hasWon.

5. Utveckling och samarbete

Ambitionsnivå: Få ett fungerande spel för godkänt. Kanske utveckla vidare i framtiden.

Tidsplan: Börja med det då jag känner att jag hinner. Satsa på labbarna för att bli klara med dom. Har ingen specifik tid projektet måste bli klart, kommer jobba med det då jag hinner.

Projektrapport

6. Implementationsbeskrivning

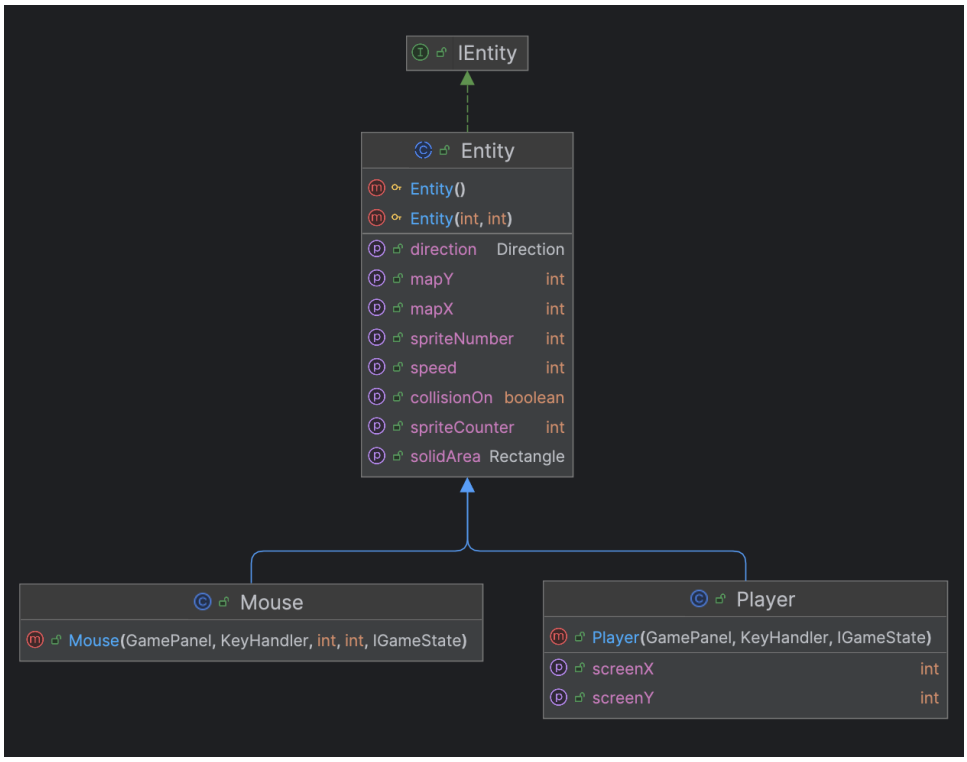
6.1. Milstolpar

Milstolpe 1–16 implementerade enligt projektplan. Milstolpe 17-19 ej implementerad.

6.2. Dokumentation för programstruktur, med UML-diagram

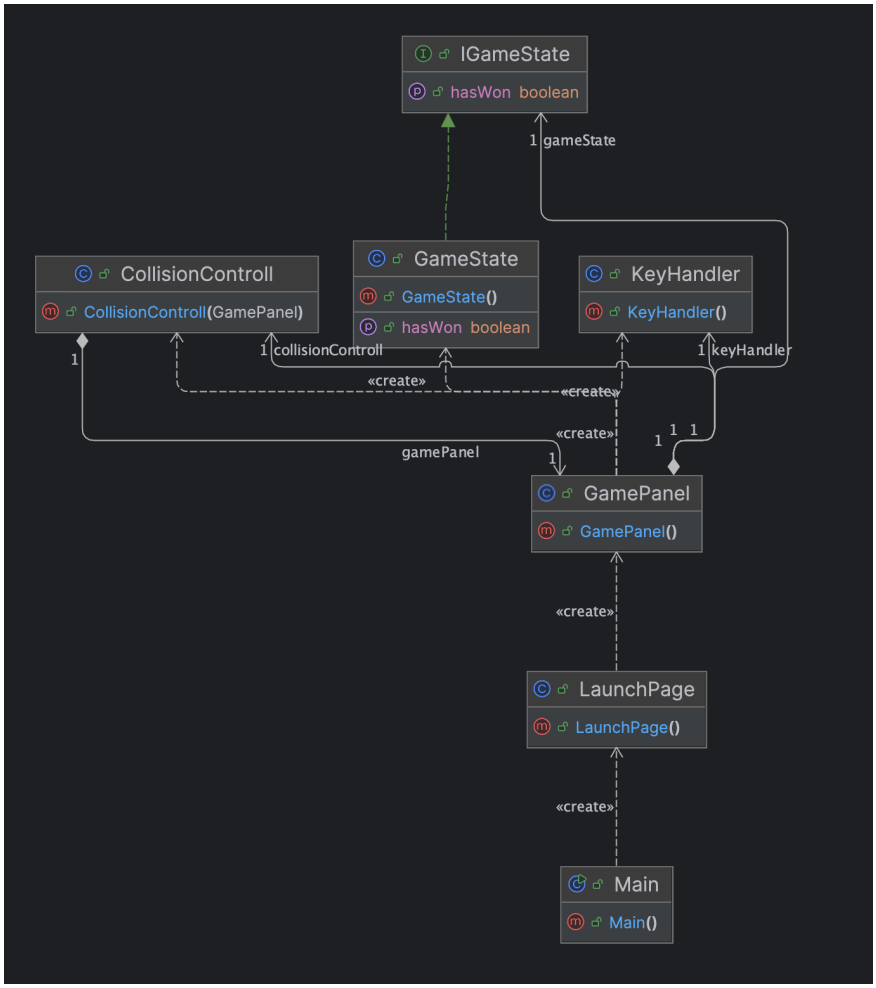
- **Övergripande programstruktur**, t.ex. att man har implementerat ett spel som styrs av timer-tick n gånger per sekund där man vid varje sådant tick först tar hand om input och gör eventuella förflyttningar för objekt av typ X, Y och Z, därefter kontrollerar kollisioner vilket sker med hjälp av klass W, och till slut uppdaterar skärmen.
  Mitt 2D-äventyrsspel handlar om en katt som jagar en mus i en tile-baserad värld. GamePanel styr spelet genom att uppdatera och rita objekt som Player (katten), Mouse och TileManager (kartan). CollisionControll hanterar kollisioner, och KeyHandler styr spelarens input. Både Player och Mouse ärver från Entity för gemensamma egenskaper. Spelloopen i GamePanel, startad via Main.java, hanterar input, uppdaterar positioner, kollar kollisioner och ritar skärmen. När katten fångar musen (kolliderar med musen) vinner katten.

- **Översikter över relaterade klasser**



Detta UML-diagram visar hur klasshierarkin för entiteterna är uppbyggd. För varje klass visas konstruktorn samt dess egenskaper. Player och Mouse representerar de två

karaktärerna i spelet, där Player och Mouse ärver den abstrakta Entity klassen som innehåller de gemensamma egenskaperna för alla entiteter. Entity klassen i sin tur implementerar IEntity intefacet och Overridear metoderna som finns i IEntity. Entity klassen fungerar som en mall och ger egenskaper som spriteNumber, speed och direction till både Mouse och Player, medan Player utökar dessa med specifika egenskaper som rörelsebilder och skärmposition. Mouse klassen hanterar musrelaterade funktioner och är kopplad till GamePanel. Denna typ av polymorfism gör att man kan återanvända kod och man får en mer lättförståelig kod.



Detta UML-diagram visar hur klasshierarkin för grafiken (det som visas på skärmen) är uppbyggd. För varje klass visas konstruktorn samt egenskaper. I denna screenshot har jag valt att fokusera på en grupp klasser relaterade till spelkörning och grafik, inklusive Main, LaunchPage, GamePanel, GameState, IGameState, KeyHandler, och CollisionControl. Detta diagram reflekterar en tydlig typhierarki där klasserna organiseras hierarkiskt för att hantera olika aspekter av spelet.

Diagrammet visar hur spelet initieras och hanterar interaktioner. Det börjar med Main, som skapar LaunchPage, som i sin tur initierar GamePanel i ett JFrame. GamePanel sätter upp GameState, som implementerar interface IGameState för att spåra tillståndet hasWon (om spelaren har vunnit) via subtypspolymorfism. GamePanel skapar även CollisionControl för kollisionskontroll och KeyHandler för tangentinput. När spelaren trycker på tangenter skickar KeyHandler uppdateringar till GamePanel, som sedan ber CollisionControl att kontrollera kollisioner. Om en kollision sker (t.ex. vinst), uppdateras GameState via setHasWon, där Overriding används för att anpassa

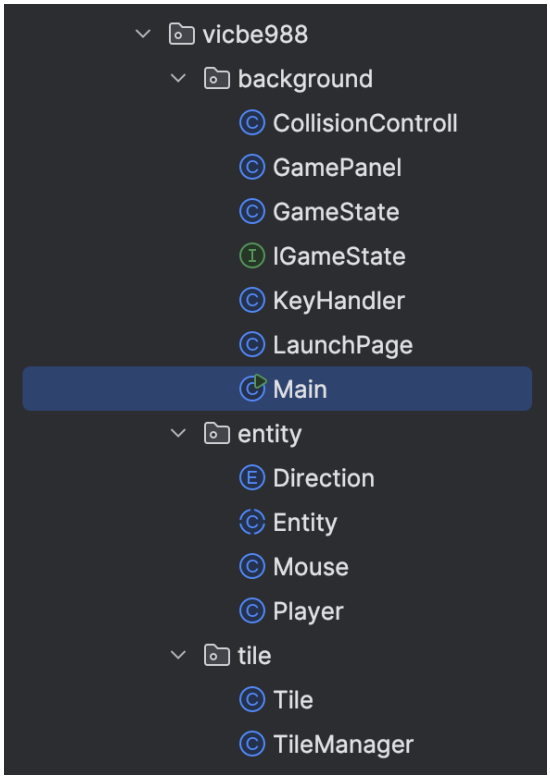
metoden efter behov. GameState fungerar som en abstrakt klass i praktiken genom att definiera gemensamma metoder via IGameState, medan inkapsling (accessnivåer) säkerställer att hasWon är privat och endast nås via getters och setters. Slutligen så renderar GamePanel spelets grafik. Alla klasser använder getters och setters.

7. Användarmanual

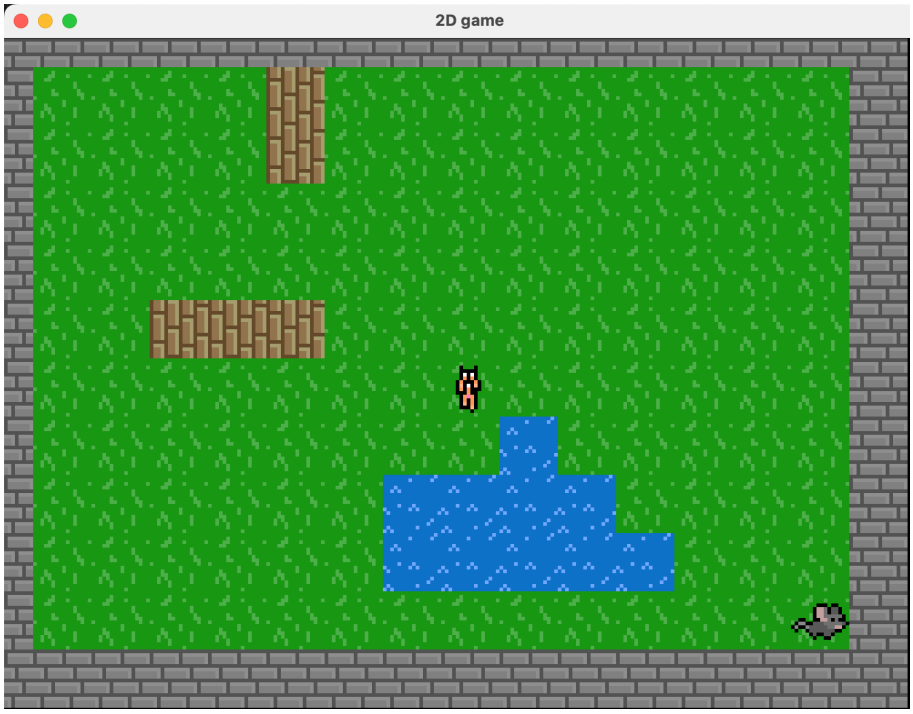
Detta är en manual för ett 2D-spel där du styr en katt som försöker fånga en mus i en värld uppbyggd av olika miljöer som gräs, vatten, trä och tegel. Spelet är utvecklat i Java och är designat för att vara lätt att komma in i, även för nybörjare.

För att starta spelet skall du göra följande:

- Öppna spelet och kör filen Main.java (i en java miljö som t.ex. IntelliJ IDEA)



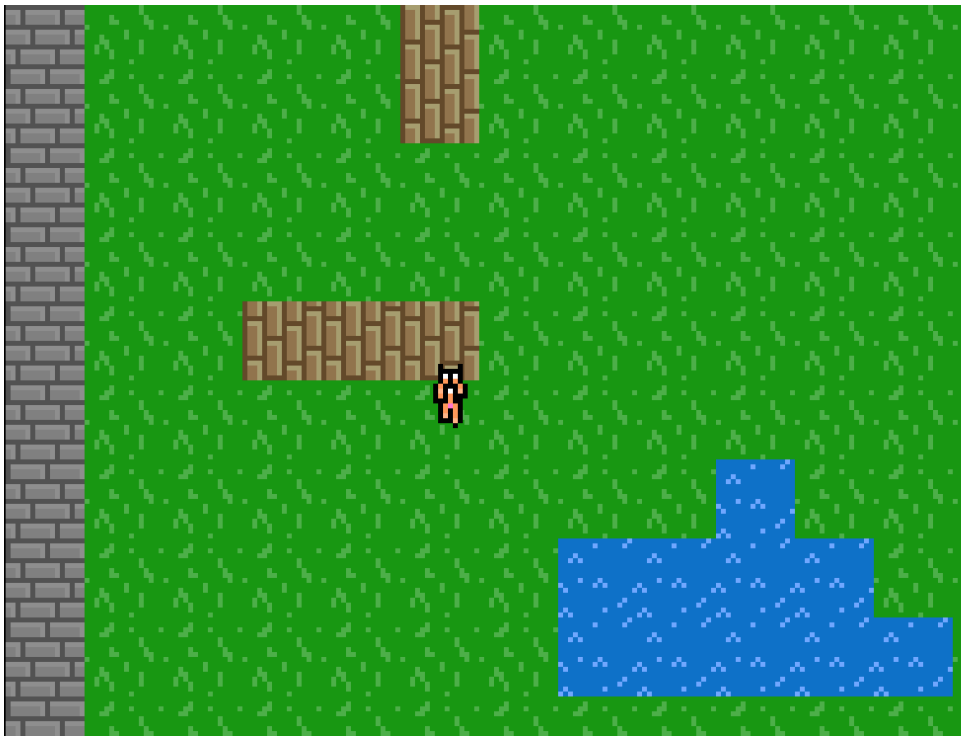
- Se till att du har java 8 eller senare installerat på din dator
- När spelat startar kommer du se en karta med en katt mitt på skärmen, då är det bara att börja spela!



Spelet är enkelt att styra och målet är tydligt. Du ska fånga musen genom att styra katten till musen. Du styr katten genom tangenterna:

- W: För att gå uppåt
- A: För att gå vänster
- S: För att gå neråt.
- D: För att gå höger.

Under spelets gång ska du undvika hinder som finns på banan. Katten kan nämligen inte gå igenom vissa typer av tiles som vatten, trä och sten.



Musen rör sig automatiskt och försöker hålla sig undan från katten. Ditt mål är att styra katten så att den kommer i kontakt med musen. När du lyckas, vinner du spelet och ett meddelande som säger "You Won!" visas på skärmen. Då kan du kryssa ned rutan med spelet.