$\pi ong!$

R&D Design Document

Door Michiel Bartens, Bryan Koch & Thijs Voncken

21-05-2013

Inleiding

πong! is een nieuwe kijk op een oud spel. Het is een videospel voor android waarin twee spelers met elkaar wedijveren om een bal in een cirkelvormige arena te houden door met hun platen langs de rand van de arena de bal te weerkaatsen. Hierbij is het de truc om de tegenstander te snel af te zijn, maar ook om hem strategisch te blokkeren; de platen kunnen immers niet door elkaar heen! Of je nou nostalgisch terugdenkt aan die goede oude Atari, of voor het eerst Pong speelt, de super snelle actie is voor iedereen! Bovendien kunnen high scores aan de wereld getoond worden met de sociale media integratie.

Verantwoording

Vergelijkbare Producten

Er bestaan tal van Pong klonen die behalve op grafisch gebied, weinig nieuws bieden. De vorm van het veld is altijd de originele rechthoek en met de stand van de plaat wordt niet geëxperimenteerd. Wij geloven dat onze nieuwe spelelementen een frisse speelervaring zullen opleveren.

Succescriteria

Vermaak

Dat dit criterium in ons voordeel is, spreekt voor zich; het doel van welk videospel dan ook is om de speler te vermaken.

Gebruikswaarde

Gebruikers kunnen hun scores op de sociale media zetten waardoor onderlinge competitie kan ontstaan. Dit verlengt de aantrekkelijkheidsduur van het spel.

Bruikbaarheid

Dit spel bevat weinig elementen die aan dit punt onderhevig zijn; de gebruikersinterface bestaat uit één statisch menu en één lijst met high scores. Daar valt weinig fout aan te gaan. De interface van het spel zelf kan in een iteratieve cyclus in de prototypes verbeterd worden.

Functionaliteit

Dit spel past goed bij Android telefoons aangezien potjes kort zijn en dus tussen andere zaken door gespeeld kunnen worden. Het is dan handig dat hier geen extra apparaat voor hoeft meegesleept te worden. Een touchscreen biedt de ideale besturing voor dit spel. Het schuiven van een plaat door deze aan te raken en vervolgens de vinger te bewegen in de gewenste richting is zeer intuïtief. Ook kan eventueel de bewegingssensor gebruikt worden. (Dit valt pas volledig te overzien wanneer een prototype gebouwd is.)

Onderhoud

Het spel zelf zal weinig onderhoud vereisen. Wel is het mogelijk om nieuwe spel modi of grafische alternatieven te creëren en deze, al dan niet tegen betaling, aan te bieden.

Requirements

Functionele eisen

- De mogelijkheid onze versie van het spel pong te spelen
- Pong tegen andere mensen kunnen spelen
- Highscores van pong bij kunnen houden

Niet functionele eisen

- Het systeem moet geen lag vertonen
- Het spel moet de regels volgen

Use cases

Use case 1: singleplayer spel spelen
Use case 2: highscore invoeren
Use case 3: app afsluiten
Use case 4: singleplayer spel starten
Use case 5: multiplayer spel starten
Use case 6: multiplayer spel spelen
Use case 7: spel afsluiten

Use case 1:

Use case naam:	singleplayer spel spelen.
Doel:	Het spelen van het spel.
Actor:	De app gebruiker.
Preconditie:	Je hebt een spel gestart.
Trigger:	use case singleplayer spel starten

Basis verloop van de events:

1. Gebruiker gebruikt knoppen om het spel te spelen.
2. Systeem geeft aan dat gebruiker geen levens meer heeft.
3 Systeem beëindigt het spel.

Alternative path:

2a gebruiker haalt een highscore.

2b. systeem beëindigt het spel.

2c Systeem start use case “highscore invoeren”

Post conditie: Het spel is afgesloten en je bent weer in het menu.

Use case 2:

Use case naam:	Highscore invoeren
Doel:	Het invoeren van je behaalde highscore
Actor:	De app gebruiker.
Preconditie:	De gebruiker heeft zojuist een highscore gehaald.
Trigger:	De gebruiker heeft zojuist een highscore gehaald.

Basis verloop van de events:

1. gebruiker voert naam in.

Post conditie:	naam, score en rang opgeslagen. Gebruiker is in het highscore venster

Use case 3:

Use case naam:	app afsluiten
Doel:	afsluiten van de app
Actor:	De app gebruiker.
Preconditie:	de app is in gebruik
Trigger:	de gebruiker gaat de app afsluiten

Basis verloop van de events:

1. gebruiker geeft aan de app af te willen sluiten.
2. Systeem vraagt om bevestiging om af te sluiten.
3. Gebruiker bevestigt de app af te willen sluiten.

Post conditie:	de app is afgesloten.

Use case 4:

Use case naam:	singleplayer spel starten
Doel:	hiermee word een spel gestart
Actor:	De app gebruiker.
Preconditie:	gebruiker bevindt zich in het menu scherm
Trigger:	gebruiker gaat een spel starten

Basis verloop van de events:

1. gebruiker geeft aan singleplayer spel te willen starten.
2. Systeem start use case singleplayer spel spelen.

Post conditie:	singleplayer spel spelen is gestart.

Use case 5:

Use case naam:	multiplayer spel starten.
Doel:	hiermee wordt een multiplayer spel gestart
Actor:	De app gebruiker.
Preconditie:	je bevind je in het menu scherm, er is netwerkverbinding met een andere app gebruiker
Trigger:	gebruiker geeft aan multiplayer spel te willen starten

Basis verloop van de events:

1. Gebruiker geeft aan multiplayer spel te willen starten.
2. Systeem laat weten te wachten op andere gebruiker.
3. Systeem geeft aan andere gebruiker te hebben gevonden.
4. Systeem vraagt om bevestiging multiplayer spel te starten.
5. Gebruiker bevestigt multiplayer spel te willen starten.
6. Systeem start use case multiplayer spel starten.

Alternative paths:

3a. systeem geeft aan geen andere gebruikers te hebben gevonden. Systeem keert terug naar het menuscherm.

5a. gebruiker wijst multiplayer spel starten af. Systeem keert terug naar het menuscherm.

Post conditie:	use case multiplayer spel spelen word gestart

Use case 6:

Use case naam:	multiplayer spel spelen
Doel:	Het spelen van het spel
Actor:	De app gebruiker.
Preconditie:	Je hebt een spel gestart.
Trigger:	use case multiplayer spel starten

Basis verloop van de events:

1. Gebruiker gebruikt knoppen om het spel te spelen.
2. Systeem geeft aan dat een van de gebruikers geen levens meer heeft.
3. Systeem geeft aan wie de winnaar is.
4. Systeem beëindigt het spel.

Alternative path:

3a. gebruiker haalt een highscore.

3b. systeem beëindigt het spel.

3c. Systeem start use case “highscore invoeren”

Post conditie:	Het spel is afgesloten en je bent weer in het menu.

Use case 7:

Use case naam:	spel afsluiten
Doel:	Het afsluiten van het spel
Actor:	De app gebruiker.
Preconditie:	er wordt een singleplayer of multiplayer spel gespeeld
Trigger:	gebruiker wil singleplayer spel of multiplayer spel afsluiten

Basis verloop van de events:

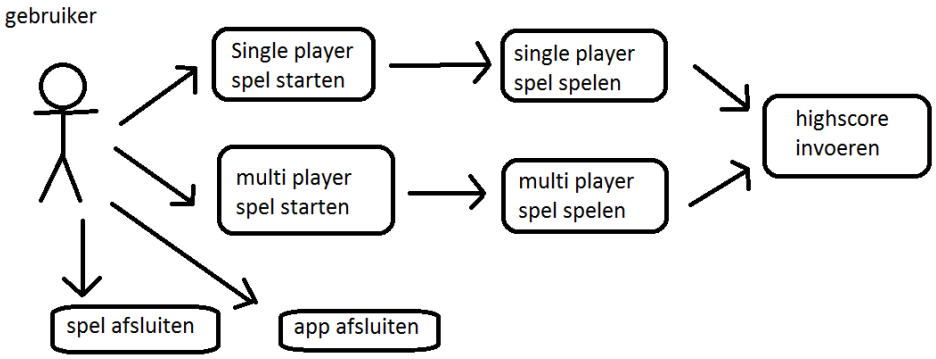
1. gebruiker geeft aan spel af te willen sluiten.
2. Syteem geeft aan dat het spel is gepauzeerd, en vraagt bevestiging om het spel af te sluiten.
3. Gebruiker bevestigt het spel af te willen sluiten.
4. Systeem beëindigt het spel.

Alternative path:

3a. gebruiker wijst spel afsluiten af. Gebruiker keert terug naar oude use case in de voormalige staat.

Post conditie:	Het spel is afgesloten en je bent weer in het menu scherm.

Use case model



Ontwerp

Globaal Ontwerp

Dit product zal worden verdeeld in verschillende componenten, die verschillende taken zullen uitvoeren om samen tot een functioneel geheel te komen.

Menu

Het eerste component is het menu, dit component zorgt ervoor dat spelers een nieuwe game kunnen beginnen. Hier moeten spelers kunnen kiezen tussen het weergeven van hun highscores en het beginnen van een single of multiplayer game.

Scoreboard

Het tweede component is het scoreboard, hier moeten spelers hun highscores kunnen zien en nieuwe highscores kunnen invoeren.

Game activity

Het derde component is de game activity, deze start alle benodigde onderdelen op om de game te kunnen laten werken. Dit component beslist ook welke input controllers er gestart moeten worden enz.

Game engine

Het vierde component is de game engine, hier gebeurt het rekenwerk en worden de bewegingen van de bal en scores van de spelers bijgehouden.

Game view

Het vijfde component is de game view, deze update vanuit de game engine het beeld dat de spelers te zien krijgen.

Game input controller

Het zesde component is de game input controller, deze vangt alle input vanuit de gebruiker op en geeft deze door aan de game engine, zodat deze kan berekenen wat er verder moet gebeuren.

Game network controller

Het zevende component wordt alleen gebruikt in het geval van een networked multiplayer game, deze onderhoudt de verbinding met het andere apparaat en zorgt er voor dat alle data goed aankomt en ontvangen wordt.

Social media hub

Het achtste en laatste component is de social media hub, deze krijgt een score doorgegeven en hoort deze dan vervolgens te posten op de correcte social media (facebook of twitter e.g.)

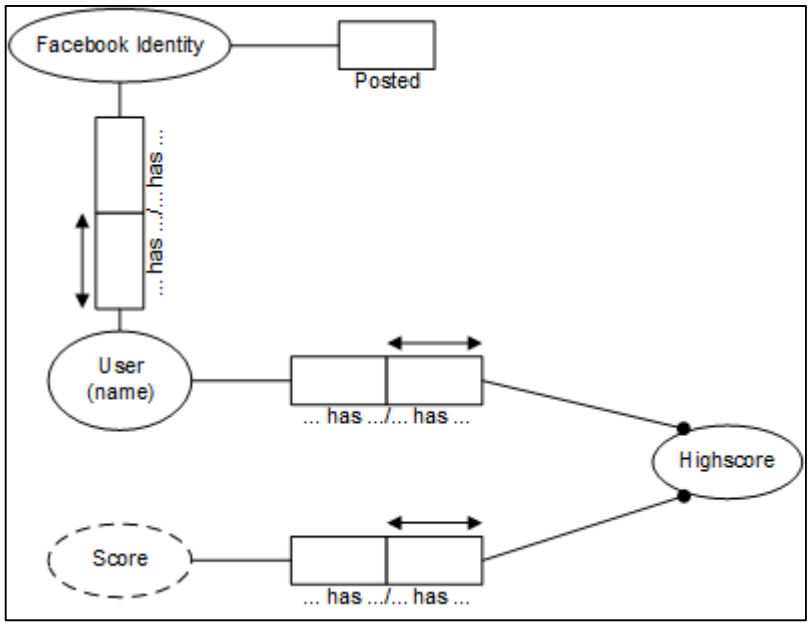
Al deze component werken nauw met elkaar samen om de app naar behoren te laten werken. Het menu start de game activity, deze start de rest van de game in de goede modus, de game engine, view en controllers werken samen door elkaar telkens te updaten om een efficiënt spel in elkaar te zetten waarvan het beeld niet bij elke zwaardere berekening vastloopt.

Vervolgens na de game werkt de engine samen met het scoreboard dat weer samen werkt met de social media hub om de scores aan de speler te kunnen representeren en op facebook te kunnen posten.

Gegevensontwerp

Om deze app goed te laten werken hebben we natuurlijk een aantal gegevens nodig. Deze gegevens worden onder andere door het systeem zelf en door de speler georganiseerd.

Ten eerste heeft de game om met facebook te kunnen synchroniseren natuurlijk inloggegevens, een API sleutel en verbindingsgegevens nodig. Vervolgens houdt de game zelf ook nog de highscores en ingevoerde namen van de lokale spelers bij om deze weer te kunnen geven.



Figuur 1. Highscore database ORM ontwerp

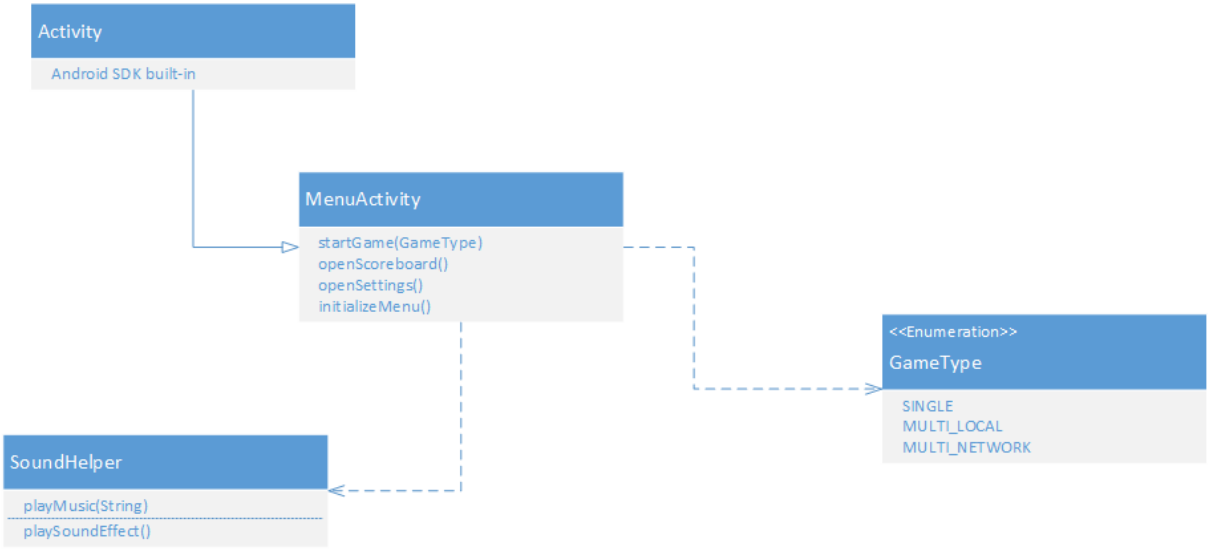
Deze gegevens wordt opgeslagen in een kleine SQL database waarin de app zijn benodigde gegevens bijhoudt over de highscores en over of ze al gepost zijn op facebook bijvoorbeeld.

Verder houdt de app natuurlijk intern alle gegevens over een lopend spel bij, posities van ballen en spelers enzovoorts, deze worden bijgehouden als interne variabelen binnen de game engine. Op het moment dat de game afgesloten wordt door een pauze event bijvoorbeeld worden deze opgeslagen in de saved instance, hierin worden het aantal levens dat elke speler heeft opgeslagen en de score van de andere speler, bovendien worden hier de posities, snelheden en versnellingen van de bal en stokken (de door spelers bestuurde objecten) opgeslagen. Indien de applicatie helemaal destroyed wordt door Android gaat ook de vooruitgang in het spel verloren.

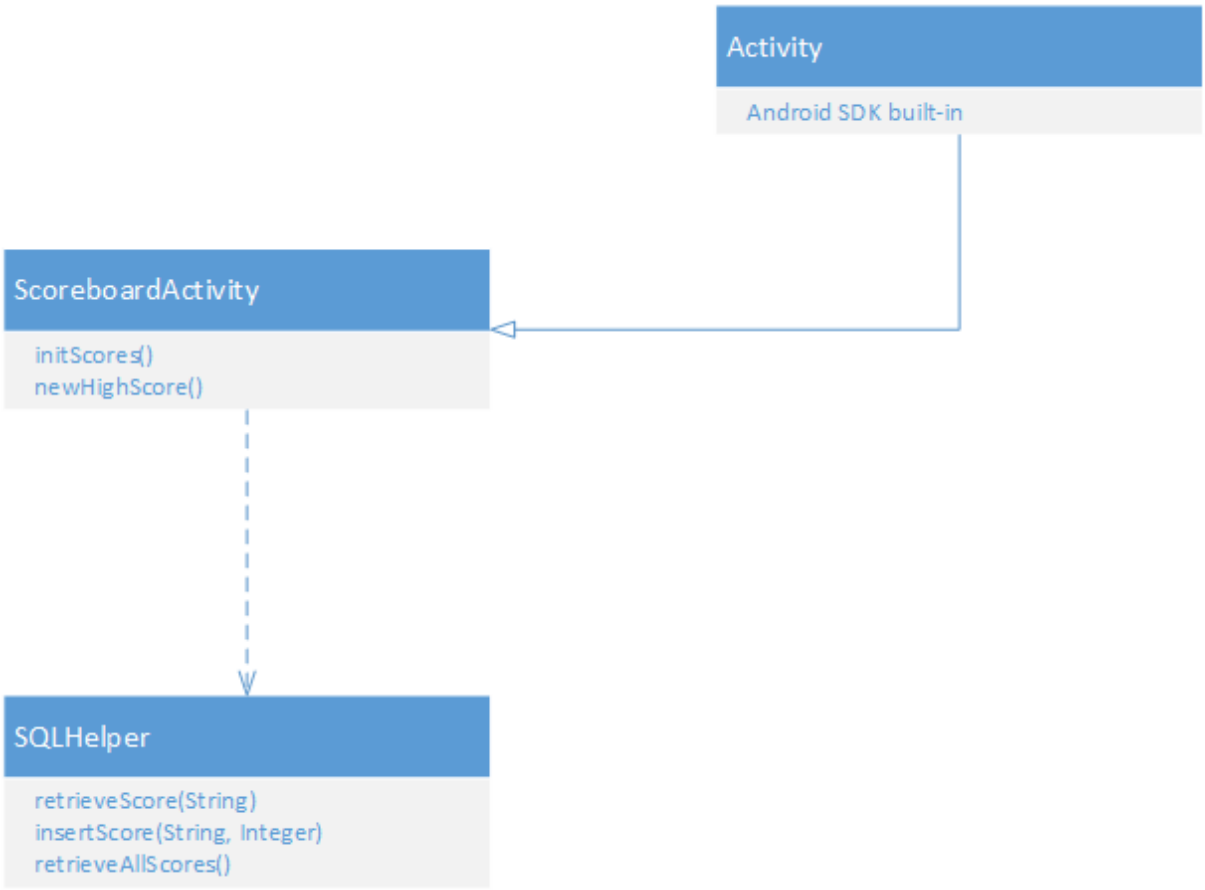
Een laatste aantal gegevens dat bijgehouden moet worden zijn de audio en video bestanden die de applicatie gebruikt als achtergrondmuziek en achtergrondafbeeldingen tijdens het menu en het spel. Deze worden in de datastructuur van de app zelf opgeslagen, het zelfde geldt voor de intern gebruikte strings enzovoort.

Detail-ontwerp per component

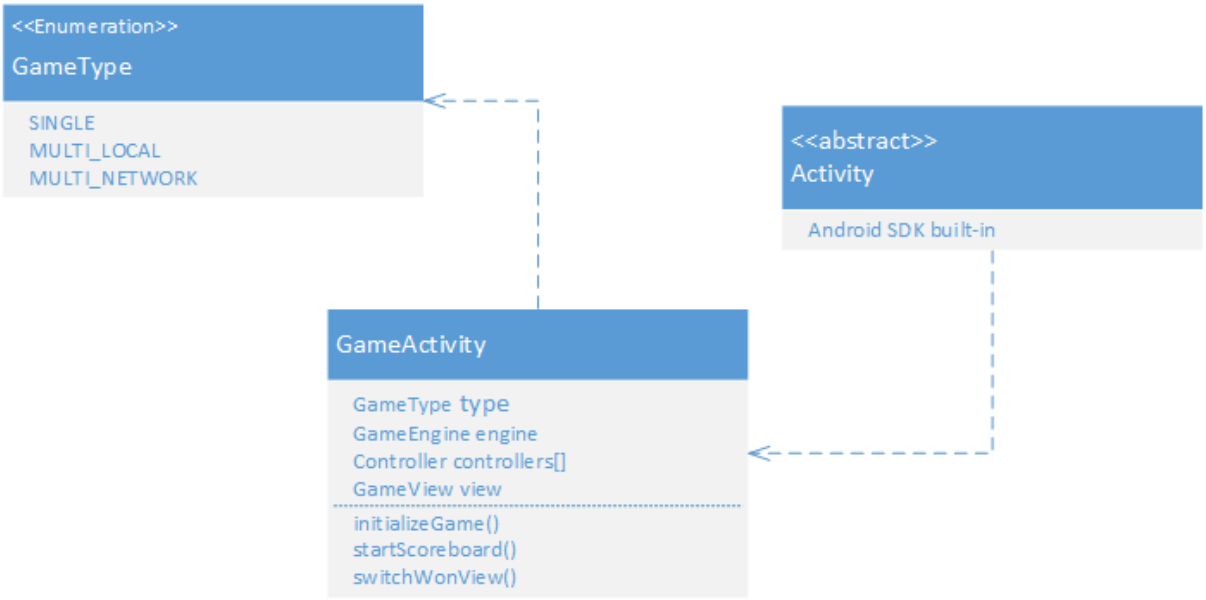
Menu



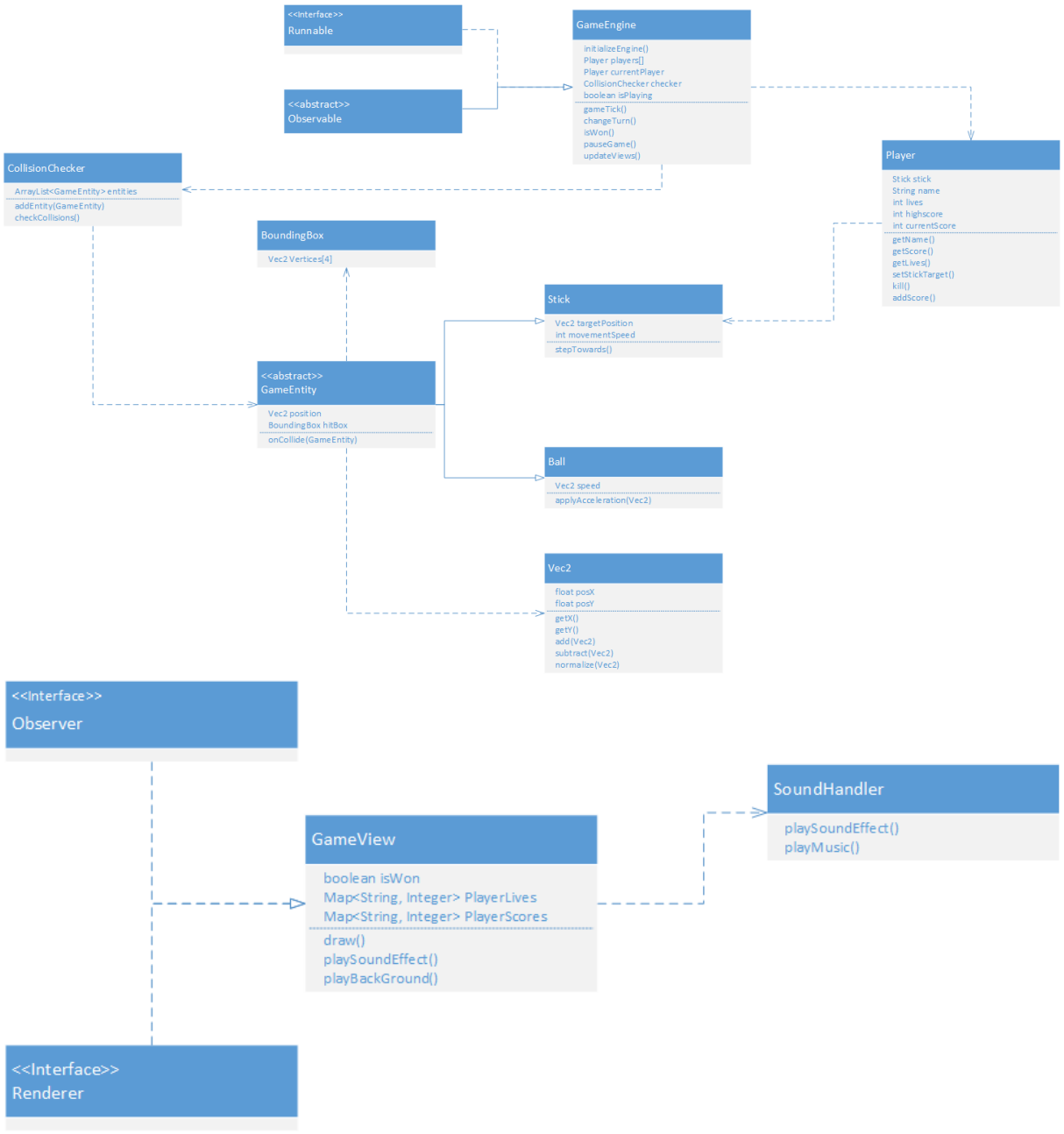
Scoreboard


Activity
Android SDK built-in
ScoreboardActivity
initScores()
newHighScore()
SQLHelper
retrieveScore(String)
insertScore(String, Integer)
retrieveAllScores()

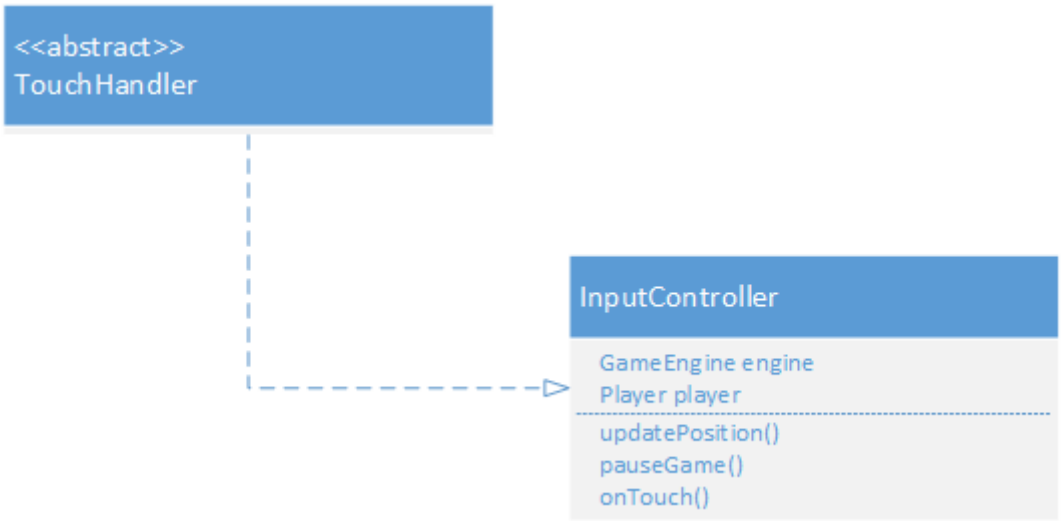
Game activity



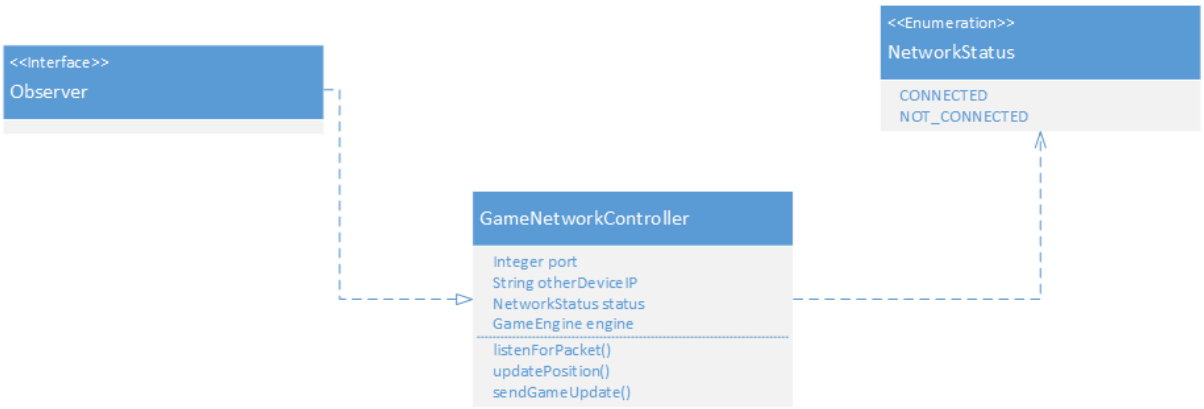
Game engine


<<Interface>>
Runnable
<<abstract>>
Observable
GameEngine
initializeEngine()
Player players[]
Player currentPlayer
CollisionChecker checker
boolean isPlaying
gameTick()
changeTurn()
isWon()
pauseGame()
updateViews()
Player
Stick stick
String name
int lives
int highscore
int currentScore
getName()
getScore()
getLives()
setStickTarget()
kill()
addScore()
CollisionChecker
ArrayList<GameEntity> entities
addEntity(GameEntity)
checkCollisions()
BoundingBox
Vec2 Vertices[4]
Stick
Vec2 targetPosition
int movementSpeed
stepTowards()
<<abstract>>
GameEntity
Vec2 position
BoundingBox hitBox
onCollide(GameEntity)
Ball
Vec2 speed
applyAcceleration(Vec2)
Vec2
float posX
float posY
getX()
getY()
add(Vec2)
subtract(Vec2)
normalize(Vec2)
<<Interface>>
Observer
SoundHandler
playSoundEffect()
playMusic()
GameView
boolean isWon
Map<String, Integer> PlayerLives
Map<String, Integer> PlayerScores
draw()
playSoundEffect()
playBackGround()
<<Interface>>
Renderer

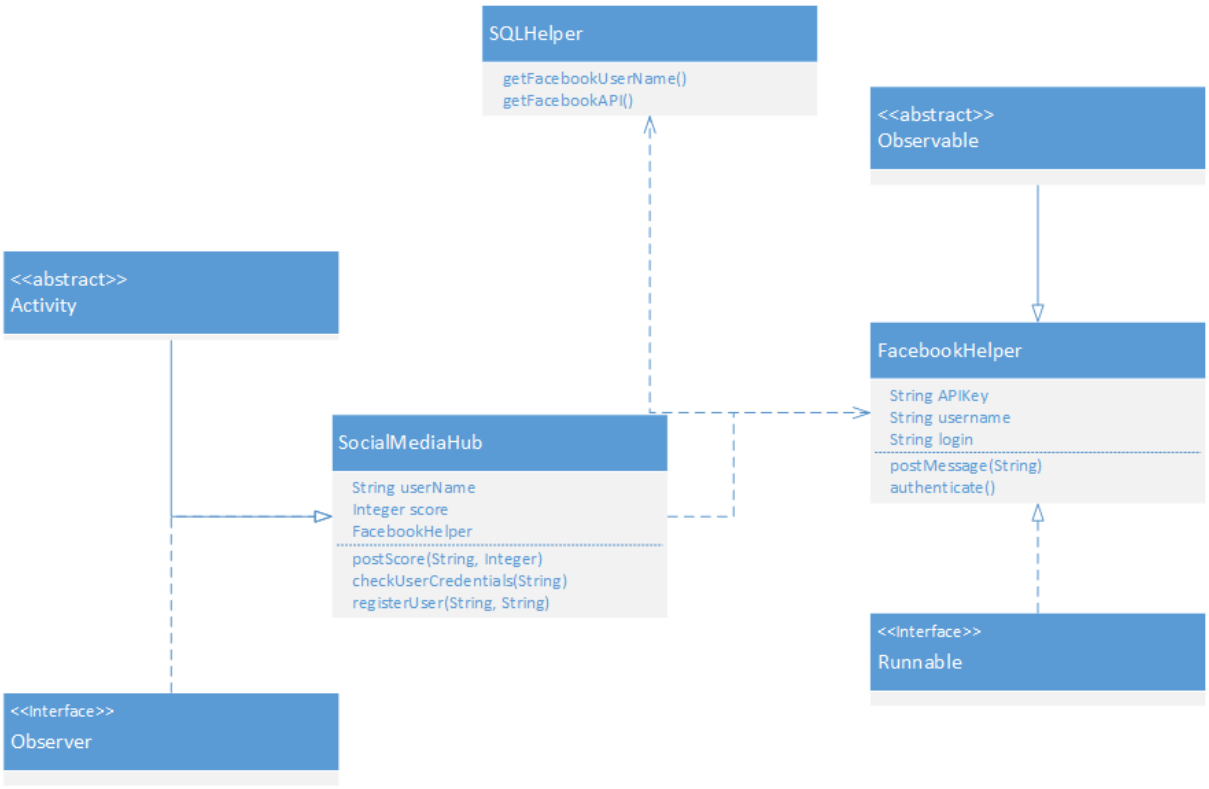
Game input controller



Game network controller



Social media hub



Gebruikersinterface