Tic´a Tac

TicTacToe

Daniel Bornstedt

25.04.18

Dette dokumentet beskriver hvordan jeg har kommet frem til applikasjonen Tic´a Tac og hvilke valg jeg har tatt gjennom utviklingen av spillet.

Applikasjonen er designet i Sketch og utviklet i Android Studio. Alle UI elementer som bilder, video, tekst og musikk kan brukes til kommersielt bruk, men kun ved å referere til kilder som opprinnelig artist eller designer.

Spillet Tic´a Tac er utviklet med disse kravene som utgangspunkt:

1. Applikasjonen skal ha en Fragment-arkitektur.
2. Applikasjonen skal gjøre bruk av et eksternt API.
3. Applikasjonen skal gjøre bruk av en lokal database.
4. Applikasjonen skal ha logikk for å kunne spille alene mot telefonen.

Vennligst les JAVADOC for en mer detaljert beskrivelse av hver klasse og metode.

Tematikk

Tic´a Tac er inspirert av kinesisk og japansk kultur. Flere av UI elementene er designet etter noe av naturen i disse landene. Derfor var det viktig for meg å ha en rød tråd gjennom hele applikasjonen. Både symboler, bakgrunner og animasjoner tar høyde for tematikken så brukeren skal føle at det er like spennende å se på, som å spille.

Målet mitt gjennom hele utviklingen var å holde 60 bilder i sekundet med 16ms “bilde tid”. Derfor har jeg fulgt Androids anbefalt design-arkitektur som for å få en stabilitet som gjør at applikasjonen også unngår “jank”. Hvis applikasjonen skipper bilder i sekundet grunnet for mye overbelasting på UI tråden så kan det oppstå.

Struktur

Applikasjonen består kun av en “activity”,og fire fragments. “activity” håndterer all media som soundPlayer og videoPlayer, men fragments håndterer UI og noe av spill logikken. Et av kravene var å bruke en fragment arkitektur i spillet så derfor har jeg tatt høyde for, og vektlagt, en som struktur som bruker lite ressurser.

Implementasjon

Fragments

Applikasjonen bruker en mainActivity.java som inneholder og bruker livssyklus logikken som kjøres utifra hva brukeren gjør i applikasjonen. Den inneholder onPause, onStop, onDestroy og onCreate. Jeg har valgt kun en activity fordi det gir oss det vi trenger av funksjonalitet og ikke minst, det gir oss stabilitet og hastighet.

I mainActivity java så lager vi nye instanser av SoundPlayer og VideoPlayer som står kan starte og stoppe video og lyd. Disse metodene kalles avhengig av hva brukeren gjør i applikasjonen.

Ved opprettelse av en activity.java så får vi også activity_main.xml filen. I denne layout filen så setter vi en "FrameLayout" som da fungerer som en container for våre fragments. Jeg har valgt å bruke ConstraintLayout som holder alt på plass. I Android forelesningen har vi lært mye om ReletiveLayouts men jeg bruker da ConstraintLayout fordi Google selv anbefaler den type layout da det gir widgets mye mer fleksibilitet.

Vår applikasjon har fire fragments. optionFragment, playerFragment, gameFragment og highScoreFragment. Alle disse fire lastes inn i frameLayouten avhenging av hva brukeren skal gjøre i appen. optionFragmentet kalles først så brukeren har mulighet til å velge forskjellige spillere eller se highScore listen.

OptionFragment står får å vise brukeren hva han/hun kan gjøre i applikasjonen. Den inneholder ImageViews, TextViews og Buttons som har tilpassede animasjoner utifra hva brukeren selv gjør. Jeg har også vektlagt Android standaren som ved å legge til en ripple effect på knappene når dem trykkes på. Trykkes det på "player1" eller "Player2" så lastes playerFragment, trykkes det på highscore, lastes highscoreFragment.

PlayerFragment inneholder TextViews der brukeren eller brukerene har mulighet til å skrive inn ønsket navn. Når dette er skrevet inn har både spiller en og spiller to mulighet til å velge et symbol i hver de 9 utvalgte ImageViews.

Når de ønskede valgene er tatt og kravene er tilfredstilt i playerFragment så blir "play" knappen tilgjengelig og man kan starte spillet.

GameFragment lastes i FrameLayout hvis "play" trykkes på. GameFragment håndterer og implementer spill logikk fra GameEngine klassen. GameFragment er hjertet av spillet. Det er her brukeren kan spille utifra om det er spiller mot CPU, eller spiller mot spiller.

Hvis highScoreFragment blir lastet inn ved trykk på highscore symbolet så får bruker oversikt over hvem som har funnet. Derfor dukker det opp en liste fra en lokal database der brukeren kan se hvem som har vunnet flest ganger. HighscoreFragment inneholder en recyclerView som inflater card_views som er anbefalt design arkitektur.

HTTP

Under utviklingen av appen så var dette kravet vanskelig fordi jeg hadde ingen anelse om hva som passet med tematikken. Derfor var det en utfordring å implementere noe som ikke stod i strid med designet. Jeg leste meg opp på dokumentasjonen til "openweathermaps" api´et, men fant fort ut at en del hadde implementert denne løsningen. Derfor gikk jeg en annen retning.

Etterhvert som appen formet seg til et ferdig produkt begynte jeg å gjøre research på flere biblioteker som Google selv anbefaler. På Android - developers så fant jeg et bibliotek som heter Volley. Grunnet tidsmessige årsaker og en sen implementering fant jeg ut at dette biblioteket passet bra da man kan enkelt gjøre en GET request mot et restAPI. Dette biblioteket kjører også "asynchronous" som vil si at den kjører på en ny tråd. Derfor sparte jeg tid da det ikke var nødvendig for en extend av AsyncTask. Å bruke AsyncTask var noe jeg selv ville bruke da det er fin øving på god Android arkitektur, men grunnet tidsmessige årsaker så unngikk jeg det.

Den implementasjonen jeg gikk for var det offisielle POKEAPI. Parseren i volley henter en url fra API´et som gir oss sprites av 900 pokemon. Dette fungerer som et easterEgg. Hvis brukeren skriver et pokemon navn i tekstfeltet i playFragment vil da den ønskede pokemon dukke opp og POKEMON THEME vil da starte.

Database

Tic´a Tac har en lokal database som inneholder et skjema med tre kolonner, PLAYER, SCORE og SYMBOL. Denne blir laget i onCreate metoden som gir oss tilgang på databasen. For å implementere data i databasen bruker jeg contentvalues. Derfor har jeg laget en egen metode som implementerer og oppdaterer databasen utifra forskjellige utkom i appen. Jeg har brukt SQLite som et rammeverk fordi det krever lite resurser og samtidig er fleksibelt.

For å vise dataen som har blitt laget i databasen har jeg laget en egen singleton klasse som henter ut dataen i en arraylist. Derfor har jeg laget en enkel instans av klassen og kan derfor populere RecyclerView med nødvendig data (JAVADOC).

Mot telefonen

Mitt ønske fra dag en var å kunne implementere en AI som gjorde at brukeren kunne justere vanskelighetsgraden selv. Dette var i følge planen på listen på ting å gjøre til sist da alt av UI var ferdig. Dessverre så var det slik at jeg ikke balanserte tiden riktig. Derfor laget jeg en randomGenerator utifra hvilke plasser som er ledig på hoved brettet. (Les JAVADOC for mer info)

Styrker

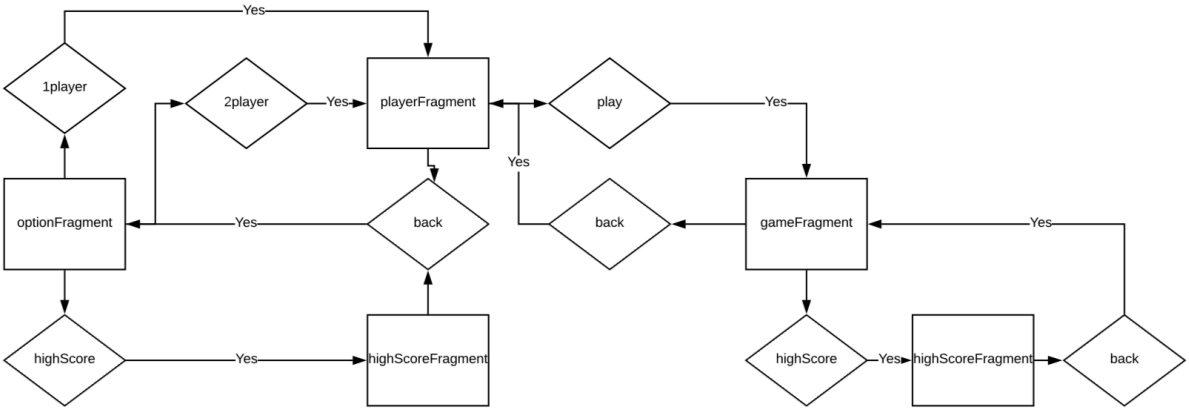
Applikasjonen jeg har laget tilfredsstiller kravene som har blitt satt. Jeg har fulgt Android design arkitekturen som gjør at applikasjonen holder 60 bilder i sekundet og jeg har ikke enda funnet jank under heavy load. I Tic´a Tac brukes det flere komponenter som feks recyclerView i highScore fragment som overgår det vi har lært i forelesningene. Jeg har også vektlagt design og animasjon for en bedre brukeropplevelse som gjør at appen er mer livlig.

Applikasjonen har også en strukturert filstruktur som gjør det lett å få en oversikt av hva klasse gjør.

Svakheter

Det jeg har konkludert med etter timesvis med jobbing er at JUNIT tester er nødvendig for store prosjekter. Applikasjonen har ikke tatt høyde for tester som burde ha vært gjort. Derfor ser jeg på dette som en svakhet da det stadig er krav i arbeidslivet. En annen svakhet er i gameFragment. Her ser jeg at jeg kunne ha refaktorert mye. Mye av logikken i gameFragment kunne ha vært over flere klasser som hjelper fragmentet og håndtere riktig logikk. Jeg har også funnet et par bugs der hvor brukerene skal skrive inn navn. Hvis brukeren lukker tastaturet før han/hun trykker på OK så kan ikke brukeren velge symbol. Dette vil jeg si er en svakhet fordi det kan være med på å ødelegge brukeropplevelsen, noe som betyr mye for meg. En annen svakhet er at databasen aldri lukkes. Jeg burde ha hatt en cursor.close/db.close så jeg ikke tapper minnet. En annen svakhet er at jeg ikke har skrevet logikk hvis spiller en og spiller to velger samme bilde.

Flyt



Ved oppstart av applikasjon blir du møt med en rød måne, tekst og fjell. Trykker du på skjermen forsvinner månen, teksten og fjellene. Du vil da kunne se tre knapper og en drage.

Du som bruker har tre alternativer, 1 player, 2 player og HighScore. Player 1 knappen bringer spiller til neste skjerm der du kan skrive inn ønsket navn etterfulgt at ønsket valg av symbol. Når dette er gjort kan man begynne spillet. I selve spillet så er det klassiske tre på råd som poenget. Får du som bruker eller CPU tre på råd vinnes det og historikken oppdateres, hvis ikke så blir det en draw. Fra denne skjermen, spilleskjermen, har også bruker tre alternativer. Settings, reset og highscore. Settings kan styre bakgrunnsmusikken, reset, restarter spillet og highscore bringer deg til highscore listen.

Java Class

HighscoreAdapter : Klasse som fungerer som en bro mellom ønsket data og vår recyclerView. Denne impleneter nødvendige metoder som vi må bruke for å populere vår liste.

CustomAlertController : Klasse som lager en custom AlertDialog.builder for å vise ved "game won" eller "draw".

BounceLogic: Klasse som lager animasjon logikken for sprett animasjonen! Her har jeg brukt noe av denne koden : **https://evgenii.com/blog/spring-button-animation-on-android/**

ButtonAnimations: Klasse som setter HardWare layout når animasjonen starter så vi kan lettere nå 60fps. Her har jeg lest Android dokumentasjonen : **https://developer.android.com/reference/android/view/animation/Animation.AnimationListener**

ImageAnimations : Klasse som lager en custom animasjon for imageviews.

SQDatabaseHelper: Klasse som oppretter en database.

GameEngine: Klasse som inneholder selve spill logikken.

TimerEngine: Klasse som inneholder tids-logikken(chronometer).

GameFragment: Fragment som inneholder og implementerer mye av GameEngine klassen.

HighscoreFragment: Fragment som inneholder recyclerView logikken.

PlayerFragment: Fragment som inneholder logikk utifra hva spilleren velger

HighScoreHolder: En view holder som vi bruker for å nå widgets i recyclerview. Her lages det også en metode som oppdatere UI utifra hvilke UI elementer recyclerviewet skal ha.

GameWin: Interface som lytter etter om spillet er funnet fra GameEngine.

RequestListener: Interface som lytter etter om vi har fått en URL fra POKEAPI.

SoundPlayer: Klasse som inneholder mediaplayer logikk for lyd.

VideoPlayer: Klasse som inneholder video logikk slik det kan spilles av lyd.

Users: En model klasse som inneholder users logikk.

SaveStates: Klasse som håndterer sharedpreferences.

RequestManager: Klasse som implementer volley biblioteket og kjører en GETrequest.

UserServices: En singletonklasse som henter data fra databasen så vi kan populere det i våres recyclerView

Layouts

activity_main:

fragment_game:

fragment_highScore:

fragment_options:

fragment_player:

my_recycler :

highscore_card:

Animations

bounce_anim:

slide_down:

slide_up:

ripple_effect:

Assets

bamboo.png : https://www.flaticon.com/free-icon/bamboo_88504#term=bamboo&page=1&position=26

bowl.png : https://www.flaticon.com/free-icon/bowl_189019

bruce.png : https://www.flaticon.com/free-icon/bruce-lee_110415#term=bruce&page=1&position=1

brush.png : laget i adobe illustrator

charmander.png : https://www.flaticon.com/free-icon/charmander_188990#term=pokemon&page=1&position=5

bulbasaur.png : https://www.flaticon.com/free-icon/bullbasaur_188989#term=pokemon&page=1&position=6

drago.png : https://www.flaticon.com/free-icon/dragon_150087#term=dragon&page=1&position=65

dragon.png : https://www.flaticon.com/free-icon/dragon_677747

drop : https://www.flaticon.com/free-icon/drop_427112#term=water%20drop&page=1&position=9

Fishes.png : https://www.flaticon.com/free-icon/chinese-carps_70867#term=carpe&page=1&position=4

koi.png : https://www.flaticon.com/free-icon/japan-koi-fish_12802#term=koi&page=1&position=3

logo.png : laget i sketch med free textFont

main_dragon : https://www.flaticon.com/free-icon/dragon_189021#term=dragon&page=1&position=7

mountains.png : redigert i AI men hentet fra : https://www.freepik.com/free-vector/snowy-mountains-illustration_765737.htm#term=mountains&page=1&position=2

music_off.png: flaticon.com

music_on.png: flaticon.com

ninja.png : https://www.flaticon.com/free-icon/ninja-portrait_13143#term=ninja&page=1&position=10

o.png: laget i sketch med teks.

x.png: laget i sketch med tekst.

panda.png : https://www.flaticon.com/free-icon/panda-bear_764064#term=panda&page=1&position=77

paper_background : https://wallpaperstock.net/old-paper-texture-wallpapers_w34279.html

Pokeball.png : https://www.flaticon.com/free-icon/pokeball_188918#term=pokeball&page=1&position=2

Sakura.png : https://www.flaticon.com/free-icon/sakura_678839#term=sakura%20flower&page=1&position=18

Shuriken.png : https://www.flaticon.com/free-icon/shuriken_486843#term=shuriken&page=1&position=36

tiger.png : https://www.flaticon.com/free-icon/tiger_208094#term=tiger&page=1&position=59

trophy : https://www.flaticon.com/free-icon/trophy-cup-silhouette_47847#term=trophy&page=1&position=9

yin : https://www.flaticon.com/free-icon/yin-and-yang_89208#term=yin&page=1&position=10

background_musc : https://www.youtube.com/watch?v=E-kY62krBjU

Kan brukes gratis så lenge man referer til skaper.

sword.wav samplet i Ableton live

Sword_two.wav Samplet i Ableton live

snowing.mov : https://videos.pexels.com videoen er også redigert i final cut pro.