Srednja šola za kemijo, elektrotehniko in računalništvo
Pot na Lavo 22
3000 Celje

GOLF S PAMETNO URO

raziskovalna naloga

Avtorji: Jon–Nik Gorenak, Tilen Goršek in Žiga Hlastec
Mentor: mag. Boštjan Resinovič
Lektorica: mag. Andreja Tkalec

Mestna občina Celje, Mladi za Celje
Celje, 2023

IZJAVA

Mentor mag. Boštjan Resinovič v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Golf s pametno uro, katere avtorji so Jon–Nik Gorenak, Tilen Goršek in Žiga Hlastec:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 4. 4. 2023

žig šole

Podpis mentorja

Podpis odgovorne osebe

»Vsak bi se moral naučiti programirati ...
saj te programiranje nauči, kako razmišljati.«
(Steve Jobs, 1995)

Zahvala

Za strokovno pomoč, nasvete in predloge se zahvaljujemo gospodu magistru Boštjanu Resinoviču, ki nas je s svojimi napotki usmerjal in tako pomagal pri nastajanju raziskovalne naloge. Zahvaljujemo se tudi gospe magistrici Andreji Tkalec, ki nam je lektorirala to nalogo. Posebej se zahvaljujemo gospodu Andražu Pušniku, ki nam je s svojimi izkušnjami, znanjem in idejami pomagal oblikovati raziskovalno nalogo.

Povzetek

Raziskovalna naloga predstavlja uporabo Apple Watch v golf igri. Z uporabo pametne ure, ki nadomesti golf palico, in računalniškega vmesnika, ki služi kot igrišče, smo simulirali golf igro. Predstavili smo naprave in aplikacije, ki so bile uporabljene pri raziskovanju, ter raziskali načine uporabe igre. Opisali smo protokole in druga orodja, ki smo jih potrebovali. V praktičnem delu smo izdelali golf igro, ki uporablja senzorje na Apple Watch in je povezana z različnimi napravami in strežniki.

Ključne besede: golf, pametna ura, Apple Watch, Unreal Engine, Swift, digitalizacija, watchOS, Xcode.

Abstract

The research thesis presents the use of Apple Watch in the game of golf. By using a smartwatch that replaces a golf club and a computer interface that serves as a playing field, we simulated a golf game. Thus, this research thesis covers the presentation of the devices and applications used in the research and also the ways of using the game that were explored along. The protocols and other tools we needed were described. In the practical part, we made a golf game that uses the sensors on the Apple Watch and is connected to various devices and servers.

Keywords: golf, smartwatch, Apple Watch, Unreal Engine, Swift, digitalization, watchOS, Xcode.

Kazalo vsebine

Kazalo slik

1 Uvod

Golf je šport, ki zahteva natančnost, vzdržljivost in koncentracijo. Čeprav se največ golf igralcev odloči za uporabo tradicionalnih orodij, kot so palice in žogice, se v zadnjem času pojavlja vse več tehnoloških rešitev, ki lahko nadomestijo ta orodja in igralcem pomagajo izboljšati njihovo igro. Eden takšnih primerov je golf igra z uporabo pametne ure.

Pametna ura je naprava, ki se lahko povezuje s pametnim telefonom in omogočajo prejemanje obvestil, spremljanje aktivnosti in uporabo različnih aplikacij. Boljše pametne ure vključujejo številne senzorje, kot so giroskop, merilnik pospeška, merilnik kisika v krvi, lokacijski senzor in podobno. Menimo, da veliko aplikacij ne izkorišča vseh možnosti, ki jih ponujajo pametne ure, zato se uporabnik znajde v dilemi: Zakaj bi kupil pametno uro, če ni popolnoma izkoriščena?

Zaradi same kompleksnosti in zahtevnosti golfa nam je bila izdelava celotne uporabniške izkušnje izziv. Bili smo v dilemi, ali se bomo lahko z golf igro, ki bo potekala z uporabo pametne ure in računalnika, približali pravi, tradicionalni golf igri. Kljub vsem izzivom in vprašanjem smo se odločili, da se bomo lotili naloge. Pri izdelavi nas je vodila strast do programiranja, oblikovanja in reševanja problemov.

2 Raziskava

2.1 Opredelitev področja in raziskovalnega problema

Golf je zelo tehnična igra, ki zahteva visoko stopnjo natančnosti, koordinacije in telesne kondicije. Golfisti se morajo spopasti z različnimi vremenskimi pogoji, dinamičnim terenom in različnimi orodji, ki jih uporabljajo pri igri. Prav tako morajo biti sposobni upravljati svoja čustva in ostati koncentrirani tudi v stresnih situacijah, kot so zadnji udarci za zmago.

Raziskovalni problem te naloge je uporaba pametne ure pri golfu: ali lahko pri golfu uporabimo pametno uro, ali lahko z njeno pomočjo nadomestimo pravo igro, ali je mogoče narediti takšno igro, da bo občutek igranja kot pri pravi igri? Sprašujemo se tudi, ali lahko z igro pomagamo golfistom, da bodo izboljšali »pravo« golf igro.

2.2 Cilji raziskovalne naloge

Cilj te raziskovalne naloge je ponuditi uporabo pametne ure Apple Watch v golf igri. S pomočjo pametne ure, ki bo nadomestila golf palico, in računalniškega vmesnika, na katerem bo tekla igra in bo deloval kot igrišče, bomo simulirali golf. S tem bomo golf igralcem ponudili alternativo h klasičnem golfu in jim pomagali pri odločitvi, ali je to tehnologija, ki bi jim lahko pomagala izboljšati igro. Prav tako bodo lahko z njeno pomočjo prihranili, saj ni potrebe po novi opremi in najemu golf igrišča. Apeliramo tudi na tiste, ki so še v dilemi, ali je golf igra za njih ali ne – brez kakršnihkoli stroškov jo bodo lahko preizkusili.

Cilji v teoretičnem delu:

- predstaviti naprave, na katerih bo tekla igra;

- predstaviti aplikacije, ki so bile uporabljene pri raziskovanju;
- predstaviti zgradbo in delovanje igre;
- raziskati in predstaviti, na kakšne načine se igra lahko uporablja.

Cilji v praktičnem delu:

- narediti golf igro s pomočjo pametne ure in računalnika;
- uporabiti različne senzorje na Apple Watch in od njih dobiti podatke;
- povezati različne naprave in strežnike;
- nuditi občutek prave golf igre;
- zapakirati funkcionalnost v odlično uporabniško izkušnjo.

2.3 Hipoteze

1. HIPOTEZA: Zaradi zaprtega Apple sistema se bomo srečali s težavami.
2. HIPOTEZA: Potrebno bo uporabiti game engine.
3. HIPOTEZA: Potrebno bo dopolniti komunikacijske protokole.
4. HIPOTEZA: Projekt presega srednješolsko znanje.
5. HIPOTEZA: Igra bo izboljšala pripravljenost golfistov na tekme.
6. HIPOTEZA: Igra bo pritegnila potencialne nove igralce golfa.
7. HIPOTEZA: Stroški projekta ne bodo presegli 200 €.

2.4 Metodologija

Z raziskovanjem smo začeli tako, da smo poiskali vire in literaturo, ki obravnavajo raziskovalno tematiko.

Sledila je preučitev teh virov in literature. Ugotavljali smo, kako si lahko pomagamo s podatki, ki jih dobimo iz pametne ure. Iskali smo možne načine povezovanje s pametno uro. Odločali smo se, na kateri platformi bo tekla

naša igra. Prebrskali smo razvijalska okolja za ustvarjanje računalniških iger, preučevali trg iger in iskali zglede.

Zatem smo poiskali in preučili orodja, aplikacije in programski jezik, ki bi jih lahko uporabljali pri delu, pretežno s praktičnega vidika. Nekateri od njih so nam že bili znani, mnogi pa še ne, zato smo veliko časa namenili njihovi adaptaciji. Uporabili smo eksperimentalno metodo, s katero smo preverili, kako različna orodja in aplikacije funkcionirajo v različnih situacijah in kaj so njihove prednosti oziroma slabosti. Na podlagi tega smo naredili izbor najljubših in dorekli, katere bomo uporabljali in katere ne. Po vseh zbranih podatkih smo oblikovali cilje in postavili hipoteze za raziskovalno nalogo.

Lotili smo se intervjuja, v okviru katerega smo ljubiteljskemu igralcu golfa predstavili našo igro in ideje. Zanimalo nas je, ali bi bil zainteresiran za igranje golfa s pomočjo pametne ure, ali bi ta lahko izboljšala njegovo igro in na kakšen način bi lahko pomagala golf igralcem.

3 Apple Watch

Apple Watch je pametna ura, ki jo je razvilo podjetje Apple. V zadnjem času je postala zelo priljubljena naprava, saj uporabnikom omogoča, da ohranijo stik s svetom in spremljajo svoje zdravje in aktivnosti. Apple Watch je bila predstavljena leta 2015 in je takoj postala priljubljena med uporabniki pametnih ur. Od takrat je podjetje Apple izdalo številne nove modele ure, s katerimi so izboljšali in dodali funkcionalnosti.

Ura ima številne funkcije, kot so spremljanje aktivnosti, merjenje srca, sporočanje in dostop do aplikacij. Apple Watch se lahko poveže s pametnim telefonom in omogoča uporabniku dostop do različnih aplikacij, kot so koledar, obvestila in glasbeni predvajalnik. Ura je opremljena s tehnologijo brezžičnega sporočanja, tako da lahko uporabnik prejema in pošilja sporočila ter klice preko ure.

Apple Watch je opremljena tudi s senzorji za spremljanje aktivnosti, ki merijo korake, prevoženo razdaljo, porabljene kalorije in druge podatke o telesni aktivnosti. Ura meri tudi srčni utrip in opozarja uporabnika, če se ta dvigne ali pade izven normalnega območja. Podpira različne povezave, kot so Wi–Fi, GPS in celularna povezava, da lahko uporabnik ostane povezan, tudi če ni v bližini pametnega telefona.

Eden od glavnih razlogov za uspeh Apple Watch je njena prilagodljivost. Uporabniki lahko namestijo različne aplikacije, spremenijo videz zaslona in uporabljajo različne pasove. Vendar pa Apple Watch ni brez pomanjkljivosti. Ena izmed glavnih težav je visoka cena, saj so nekateri modeli zelo dragi. Ura tudi ni zelo avtonomna, saj se mora pogosto polniti, kar lahko predstavlja težavo za tiste, ki se pogosto znajdejo na poti.

Vseeno pa je Apple Watch zelo priljubljena naprava in verjetno je, da bo njena priljubljenost v prihodnje še naraščala. Uporabniki cenijo njeno

prilagodljivost, zdravstvene funkcije in sposobnost spremljanja aktivnosti, zato je to odlična izbira za tiste, ki iščejo pametno in zdravstveno usmerjeno uro.

3.1 Apple Watch Series 3

Apple Watch Series 3 je tretja generacija pametne ure, ki jo je razvila družba Apple in je bila predstavljena septembra 2017. Opremljena je z zmogljivo tehnologijo, ki omogoča številne funkcije, kot so merjenje aktivnosti, spremljanje zdravja, sprejemanje klicev in sporočil, spremljanje časa in dostop do aplikacij.

Ena glavnih novosti Apple Watch Series 3 je vključitev brezžičnega interneta. Ura lahko namreč deluje neodvisno od pametnega telefona in omogoča dostop do interneta prek Wi–Fi ali mobilnega omrežja. To omogoča uporabo aplikacij in storitev, kot so spletni brskalnik, elektronska pošta in spletne aplikacije, ne da bi bilo potrebno v bližini imeti pametni telefon.

Opremljena je z zmogljivim procesorjem in vgrajenim GPS–om, ki omogočata hitro delovanje in natančno spremljanje lokacije. Pametni zapestni pas ima tudi vodotesno zasnovo in lahko meri številne kazalnike zdravja, kot so srčni utrip, koraki in poraba kalorij.

Apple Watch Series 3 je na voljo v različnih barvah in materialih, kot so aluminij, nerjaveče jeklo in keramika. Ima tudi številne vrste pasov, tako da lahko vsak uporabnik izbere tistega, ki mu najbolj ustreza.

Slika 1: Apple Watch Series 3.
Vir: https://store.storeimages.cdn-apple.com/4982/as-images.apple.com/is/FQKV2?wid=1673&hei=1353&fmt=jpeg&qlt=95&.v=1517334312067

3.2 Apple Watch SE

Apple Watch SE je pametna ura, ki jo je razvila družba Apple in je bila predstavljena septembra 2020. Opremljena je z zmogljivo tehnologijo, ki omogoča številne funkcije, kot so merjenje aktivnosti, spremljanje zdravja, sprejemanje klicev in sporočil, spremljanje časa in dostop do aplikacij. Ima zmogljiv procesor, vodotesno zasnovo in lahko meri številne kazalnike zdravja.

Apple Watch SE je na voljo v različnih barvah in materialih, kot so aluminij, nerjaveče jeklo in keramika. Ima tudi številne vrste pasov.

Eden od glavnih razlik med Apple Watch SE in drugimi modeli pametnih ur je cena. Apple Watch SE je namreč cenovno dostopnejša od drugih modelov in kljub nižji ceni ponuja številne zmogljive funkcije za merjenje aktivnosti,

spremljanje zdravja in dostop do aplikacij in storitev, kot jih ponujajo dražji modeli Apple Watch.

Slika 2: Apple Watch SE.
Vir: https://www.mimovrste.com/i/51057013/550/550

3.3 Apple Watch Series 7

Apple Watch Series 7 je sedma generacija pametne ure, ki jo je razvila družba Apple in je bila predstavljena septembra 2021. Opremljena je z zmogljivo tehnologijo, ki omogoča številne funkcije, kot so merjenje aktivnosti, spremljanje zdravja, sprejemanje klicev in sporočil, spremljanje časa in dostop do aplikacij.

Glavna novost Apple Watch Series 7 je vključitev funkcije merjenja kisika v krvi. Na ta način lahko opozori uporabnika, če se kisik zniža na nezdravo raven. To lahko pomaga ljudem, ki trpijo zaradi bolezni, kot so astma in pljučna bolezen, da spremljajo svoje zdravje in se posvetujejo z zdravnikom, če je potrebno.

Kot prejšnji modeli je tudi ta opremljen z zmogljivim procesorjem, ima vodotesno zasnovo in lahko meri številne kazalnike zdravja, kot so srčni utrip, koraki in poraba kalorij.

Apple Watch Series 7 je na voljo v različnih barvah in materialih, kot so aluminij, nerjaveče jeklo in keramika. Ima tudi številne vrste pasov.

Slika 3: Apple Watch Series 7.
Vir: https://www.mimovrste.com/i/69080170

4 Xcode

4.1 Osnove

Xcode je Applovo integrirano razvojno okolje (IDE) za operacijski sistem macOS, ki je bilo izdano leta 2003. Program je brezplačen za vse uporabnike macOS in je na voljo preko aplikacije App Store. Razvijalcem omogoča, da ustvarijo programe za Applove operacijske sisteme: macOS, iOS, watchOS, tvOS in iPadOS. Podpira številne programske jezike, kot so C, C++, Objective–C, Objective–C++, Java, AppleScript, Python, Ruby, ResEdit in Swift.

Znotraj Xcode–ovega okolja so na voljo številna orodja, kot so urejevalnik kode, upravljalnik različic kode, razhroščevalnik, testiranje in še veliko več. Ponuja tudi orodje za razvoj uporabniškega vmesnika aplikacij, kar omogoča razvijalcem, da vizualno oblikujejo in preizkusijo izgled in funkcionalnost svojih aplikacij.

Poleg tega Xcode omogoča ogromno drugih orodij in funkcij, ki razvijalcem pomagajo, da ustvarijo kakovostno aplikacijo, vključuje na primer orodja za uporabo storitve oblaka Apple CloudKit, napredno izdelavo grafičnih uporabniških vmesnikov z vmesnikom Auto Layout, uporabo Applovih ogrodij (frameworks) in orodij za deljenje kode z drugimi razvijalci.

Vse to orodje in funkcije v Xcode–u razvijalcem pomagajo pri lažjem, hitrejšem in bolj učinkovitem ustvarjanju kakovostne programske opreme za različne naprave in platforme podjetja Apple.

4.2 Uporaba

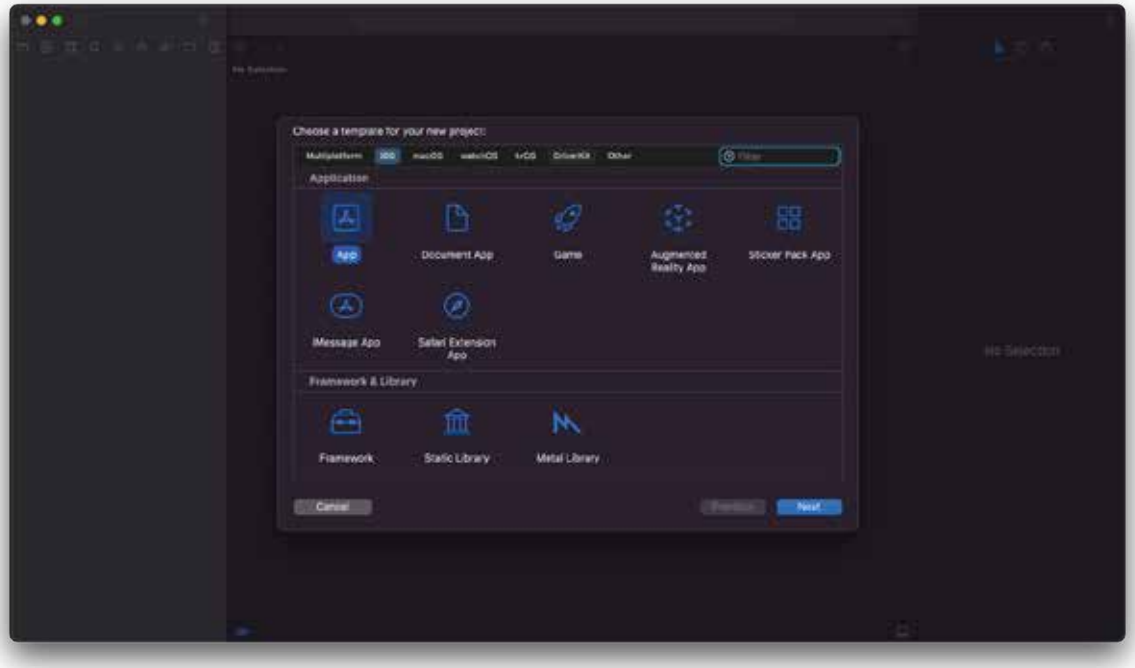
4.2.1 Ustvarjanje novega projekta

Ko odpremo Xcode, se prikaže pozdravni zaslon, na katerem so ponujene naslednje možnosti: ustvarjanje novega projekta (na primer za iPhone, iPad ...), kloniranje in odpiranje obstoječega projekta ali datoteke. V desnem meniju so prikazani nedavno uporabljeni projekti.

Slika 4: Pozdravni zaslon v Xcode.

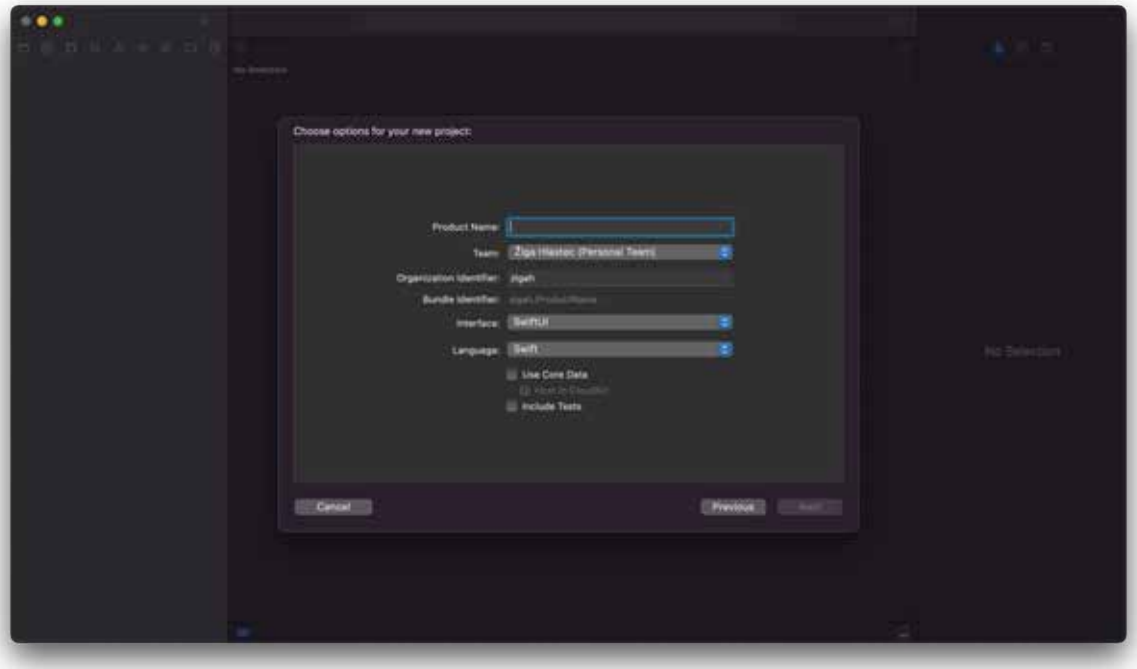
Če želite ustvariti nov projekt, morate izbrati možnost »Create a new Xcode project«. Prikaže se zaslon, kjer bo izbrana predloga za nov projekt. V zgornjem meniju lahko izberemo platformo. Izbiramo lahko med različnimi platformami: iOS, macOS, watchOS, tvOS, DriverKit in drugim. Pod oknom aplikacije lahko izberemo številne možnosti: aplikacijo, aplikacijo za dokumente, igro, aplikacijo za obogateno resničnost, aplikacijo za paket nalepk, aplikacijo iMessage in aplikacijo za razširitev Safari. V drugem oknu

z imenom »Framework and Library« lahko izberemo Framework, Static Library ali Metal library.



Slika 5: Izbiranje predloge.

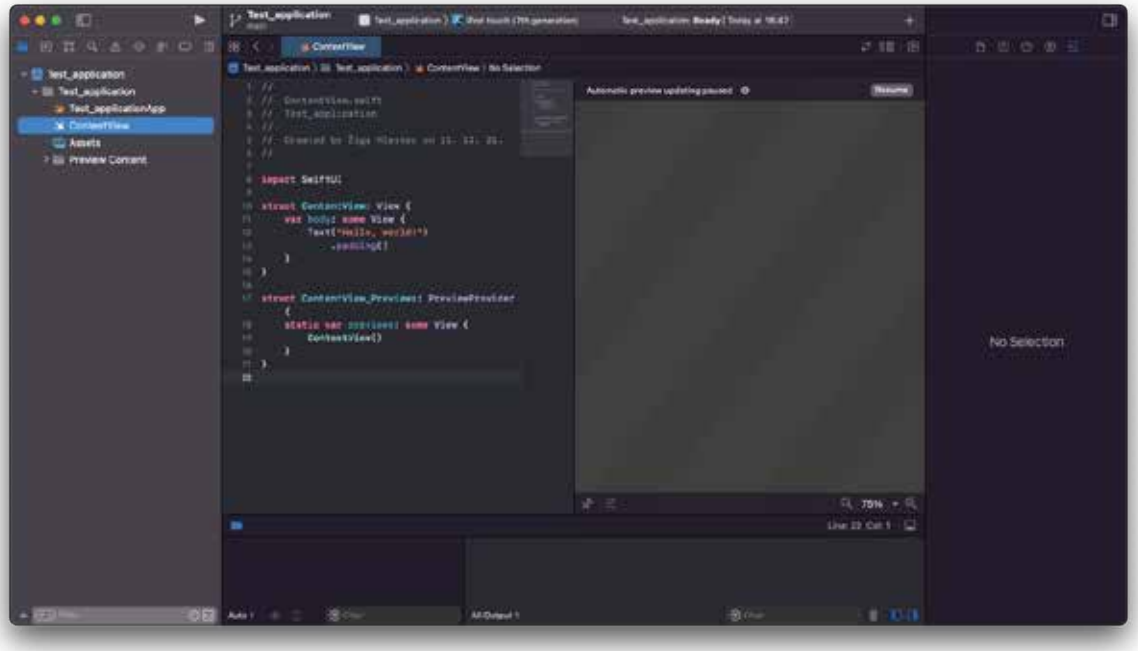
Ko izberemo predlogo, lahko na naslednjem zaslonu konfigurirate številne možnosti: ime izdelka, ime vašega projekta, ekipo, identifikator organizacije, vmesnik, kjer je mogoče izbrati SwiftUI ali Storyboard, in programski jezik, ki se bo uporabljal za programiranje.



Slika 6: Konfiguracija novega projekta.

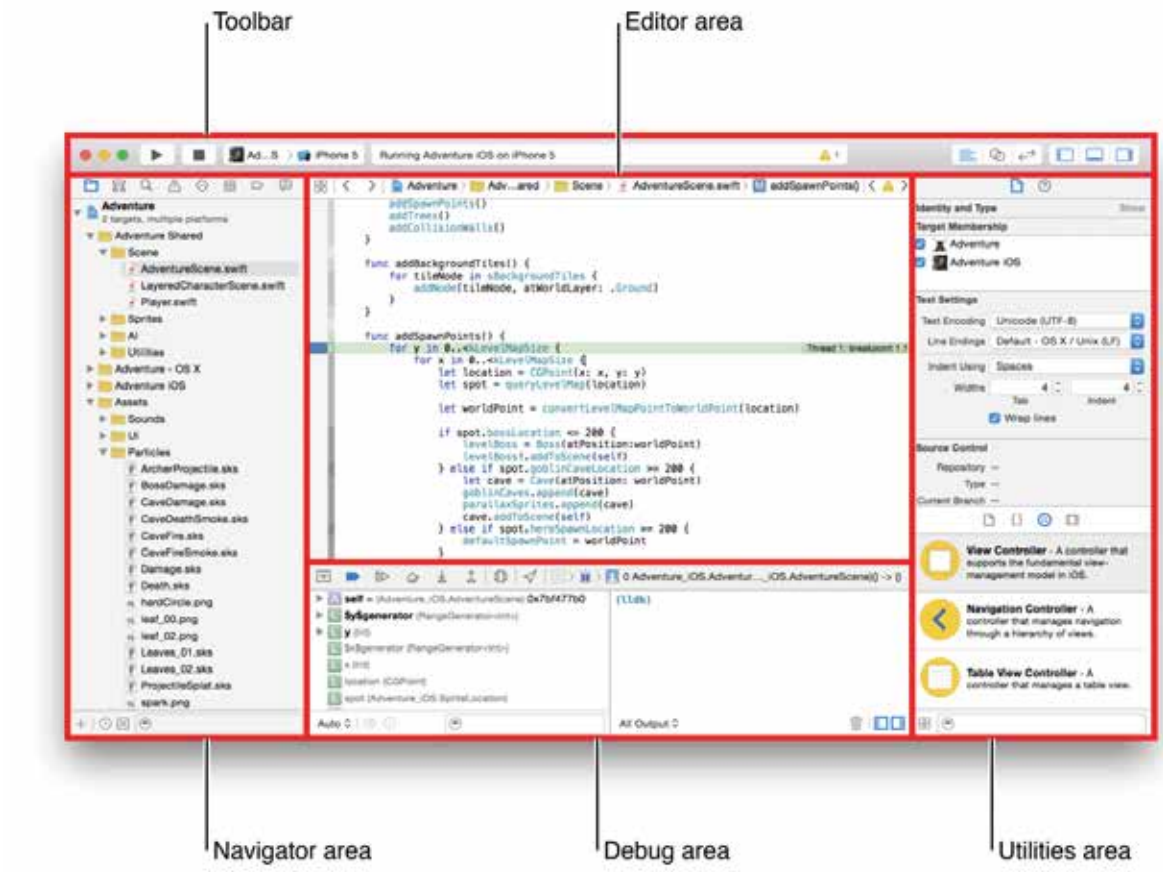
4.2.2 Okno delovnega prostora

Ko ustvarimo nov projekt, bo začetni zaslon videti kot na spodnji sliki. Ta zaslon se imenuje okno delovnega prostora Xcode in je primarni vmesnik za ustvarjanje in upravljanje projektov. Uporabnik ga lahko prilagodi. Privzeto je ustvarjena aplikacija Hello world.



Slika 7: Privzeta Hello world aplikacija.

Okna delovnega prostora so sestavljena iz petih glavnih komponent: orodna vrstica, območje urejevalnika, območje za krmarjenje, območje za odpravljanje napak in območje pripomočkov.



Slika 8: Komponente delovnega okolja.

Vir:

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_WrkspaceWindow_2x.pngOrodna vrstica

Na vrhu delovnega prostora je orodna vrstica, ki omogoča hiter dostop do pogosto uporabljenih ukazov. Gumb »Run« se uporablja za izvajanje kode. Z gumbom za zaustavitev lahko kodo, ki se izvaja, prekinete. Napravo, na kateri bo emulirana koda, lahko izberete v meniju sheme. Pregledovalnik dejavnosti prikazuje napredek različnih nalog in morebitne težave z našo kodo, na primer napake in opozorila. Območje urejevalnika je mogoče konfigurirati z gumbi za konfiguracijo urejevalnika. Gumb za konfiguracijo delovnega prostora skrije ali prikaže izbirna področja: navigator, odpravljanje napak in pripomočki.

Slika 9: Orodna vrstica.

Vir:

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_Toolbar_2x.png

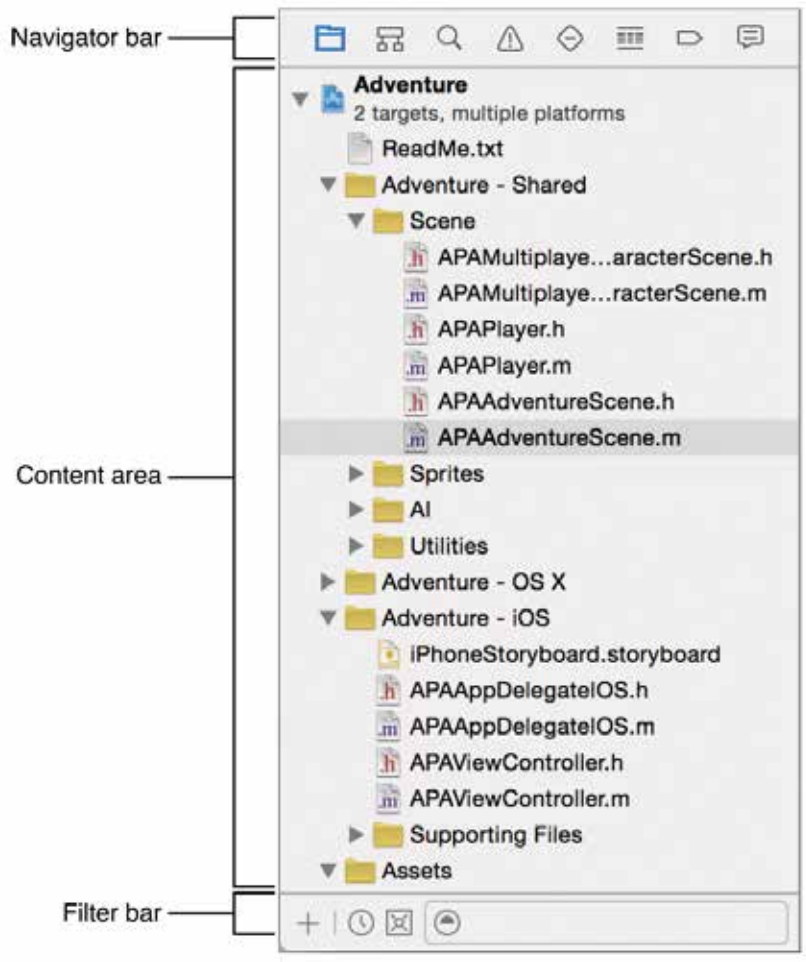
4.2.2.1 Navigacijsko območje[1]

V navigacijski vrstici lahko izberemo številne možnosti:

- **Navigator projekta:** Dodajajmo, odstranjujmo, združujemo in drugače upravljamo datoteke v svojem projektu ali pa izbiramo datoteko, da si ogledamo ali spreminjamo njeno vsebino.
- **Krmar po simbolih:** Kot seznam ali hierarhija se pomikamo skozi simbole v našem projektu. Omejujemo prikazane simbole na kombinacijo samo razredov in protokolov, samo simbolov v našem projektu ali samo vsebnikov z uporabo gumbov na levi strani vrstice filtrov.
- **Iskanje navigatorja:** Če želimo odkriti katerikoli niz v našem projektu, lahko uporabimo možnost iskanja in filtre.
- **Krmar po težavah:** Ko odpremo, analiziramo in zgradimo svoj projekt, lahko naletimo na težave, opozorila in napake. Prikazani bodo tu.
- **Testni navigator:** Tu lahko ustvarimo, upravljamo, izvajamo in pregledujemo teste.

[1] Povzeto po: https://developer.apple.com/documentation/xcode/

- **Navigator za odpravljanje napak:** V določenem trenutku med izvajanjem programa je tu mogoče pregledati tekoče niti.
- **Navigator prekinitvenih točk:** Prekinitvene točke je mogoče natančno nastaviti z definiranjem atributov, kot so sprožilne situacije.
- **Navigator poročil:** Oglejmo si zgodovino svojih nalog, kot so izgradnja, zagon, odpravljanje napak, stalna integracija in nadzor vira.



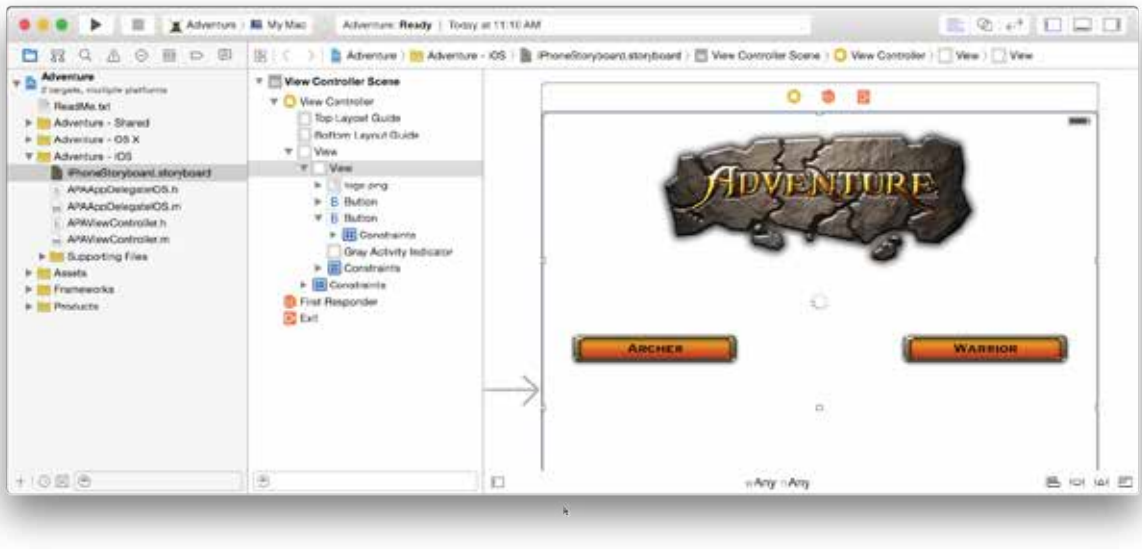
Slika 10: Navigacijska vrstica.

Vir:

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_Navigator_area_2x.png

4.2.2.2 Območje urejevalnika

Večina razvojnega dela poteka v urejevalniku, ki je vedno viden v oknu delovnega prostora.



Slika 11: Območje urejevalnika.

Vir:

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/IB_H_selected_ib_file_2x.png

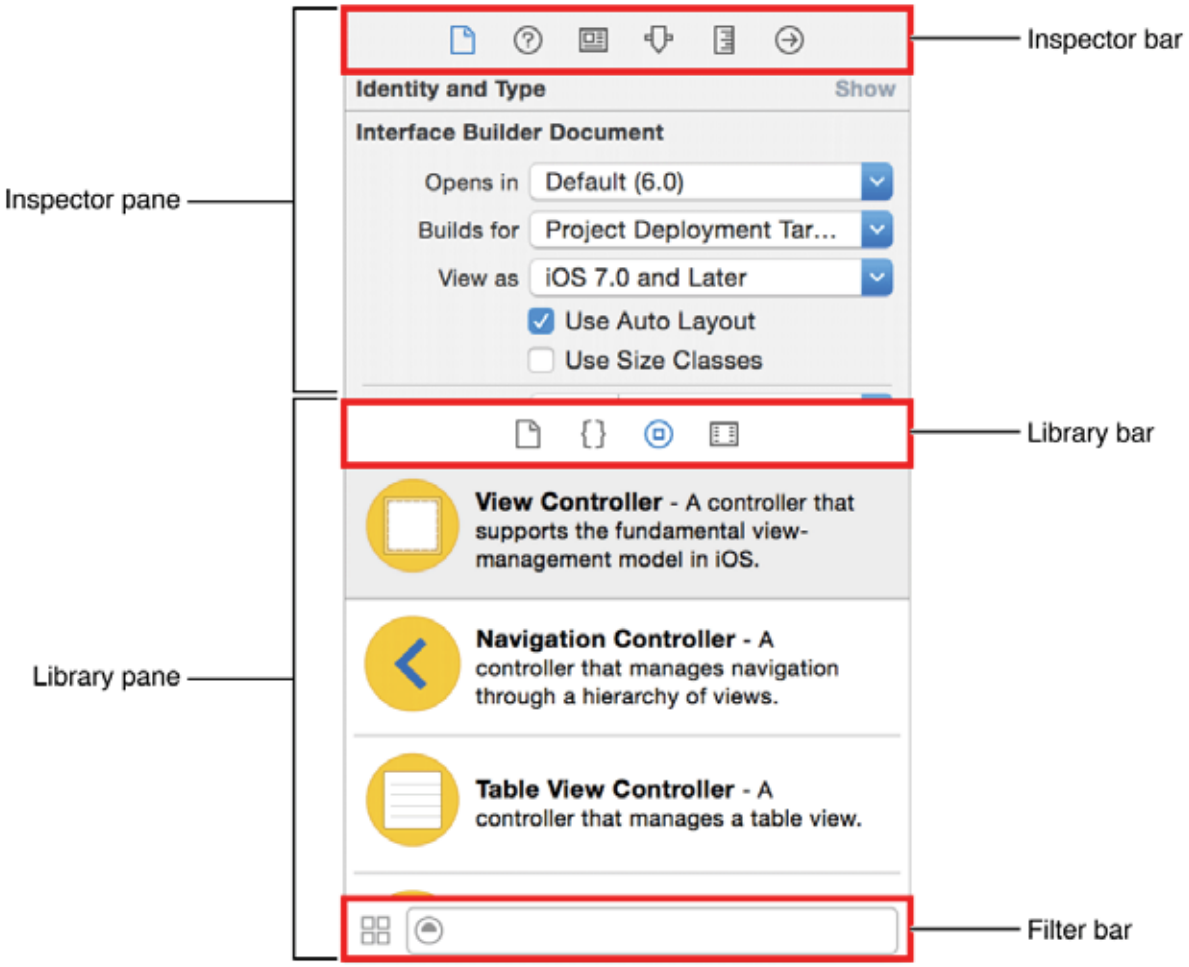
4.2.2.3 Orodja[2]

Nahajajo se v polju pripomočkov na desni strani okna delovnega prostora.

V zgornji vrstici je izbrano orodje za določeno nalogo. V vrstici sta običajno vidni dve:

- **Pregledovalnik datotek:** Tu si lahko ogledamo in upravljamo metapodatke za izbrano datoteko.
- **Hitra pomoč:** Tu si lahko ogledamo podrobnosti o elementih.

[2] Povzeto po: https://developer.apple.com/documentation/xcode/



Slika 12: Orodna vrstica.

Vir:

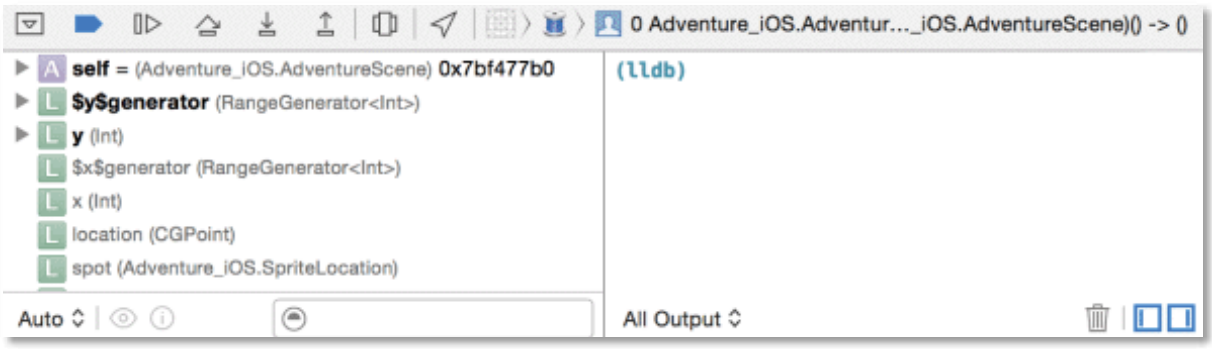
https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_Util_area_2x.png

V knjižnični vrstici lahko dostopamo do knjižnic projekta:

- **Predloge datotek:** Predloge za priljubljene oblike datotek.
- **Delčki kode:** Kratki delčki izvorne kode, ki jih lahko vključimo v svoj projekt.
- **Predmeti:** Elementi uporabniškega vmesnika za našo aplikacijo.
- **Mediji:** Grafika, ikone, glasbene datoteke in druge podobne stvari.

4.2.2.4 Območje za odpravljanje napak

Tu lahko preizkusimo svojo kodo in odpravimo morebitne napake. Glavni namen razhroščevalnika je omogočiti programerju, da sledi dejavnostim ciljnega programa v izvajanju in spremlja spremembe v računalniških virih, ki lahko signalizirajo napačno kodo.



Slika 13: Območje za razhroščevanje.

Vir:

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_WrkspaceWindow_2x.png

5 Swift

5.1 Osnove

Swift je objektno orientiran programski jezik, ki ga je razvila družba Apple za razvoj aplikacij za svoje platforme, kot so iOS, iPadOS, macOS, watchOS in tvOS. Eden od glavnih ciljev pri razvoju jezika je bila enostavnost uporabe. Jezik je zasnovan tako, da je čim bolj pregleden in enostaven za branje, kar omogoča hitrejši razvoj aplikacij. Izdan je bil leta 2014 in se je hitro razširil med razvijalci aplikacij, saj je enostaven za uporabo in zelo funkcionalen. (Buttfield-Addison, Manning, & Nugent, 2017)

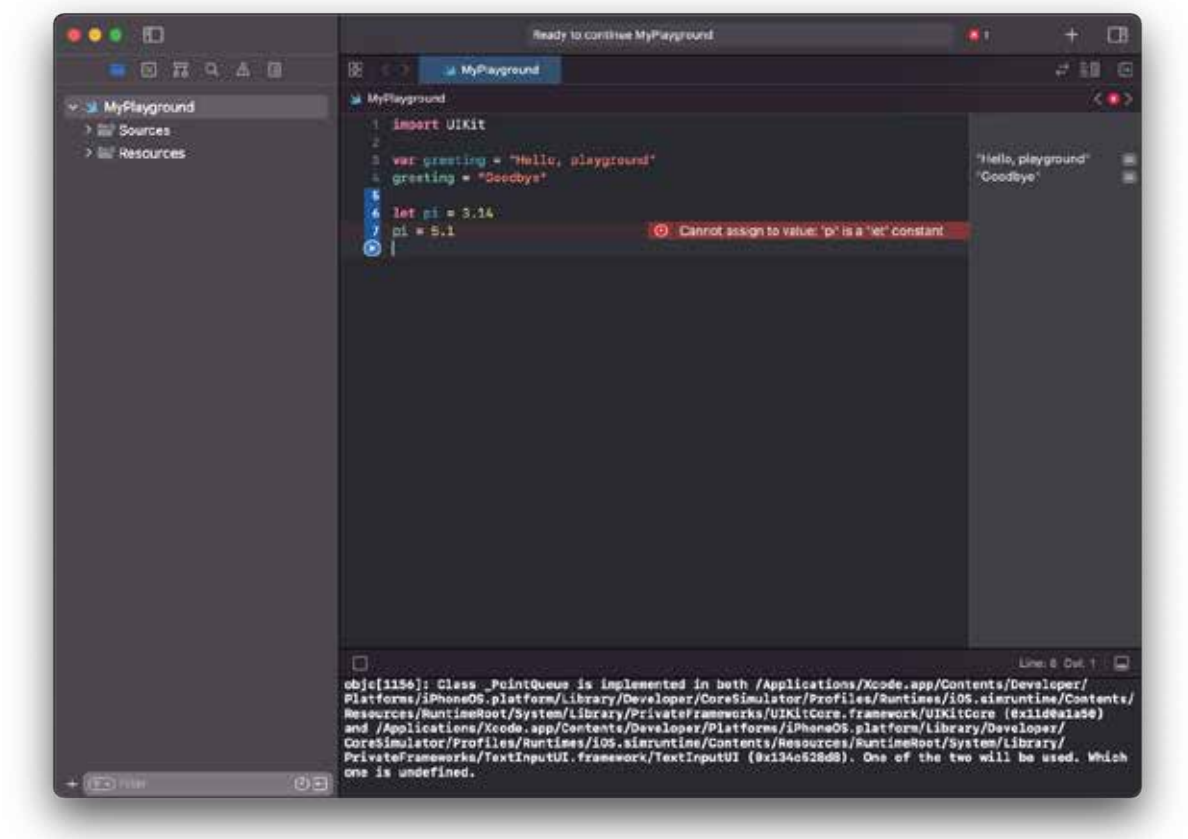
Swift je zelo zmogljiv programski jezik, saj omogoča razvijalcem, da v njem izdelujejo zelo zahtevne aplikacije, ki delujejo hitro in učinkovito. Prav tako je tudi zelo prilagodljiv, saj omogoča, da razvijalci združijo kode iz drugih jezikov, kot sta Objective–C in C++. To razvijalcem omogoča, da izkoristijo svoja obstoječa znanja in jih uporabijo pri razvoju aplikacij za platforme podjetja Apple. Swift je tudi zelo varnostno usmerjen jezik, saj ima učinkovite mehanizme za preprečevanje napak in varnostnih ranljivosti. To je še posebej pomembno pri razvoju aplikacij za mobilne naprave, saj lahko uporabnikom povzročijo resne težave. (Altexsoft, 2021)

5.2 Sintaksa

5.2.1 Spremenljivke

V Swiftu se za ustvarjanje spremenljivk uporabljata »let« in »var«. »Let« služi pri ustvarjanju nespremenljivih spremenljivk (konstant), medtem ko »var« pomaga pri ustvarjanju spremenljivih spremenljivk. Oba ustvarjata spremenljivke, ki vsebujejo sklic ali vrednost.

Razlika med njima je v tem, da je konstanti (ustvarjeni z uporabo »let«) treba nekaj dodeliti, preden jo prvič uporabimo, kasneje pa ni mogoče spreminjati njene vrednosti. Ko je spremenljivka deklarirana z uporabo »var«, ji je mogoče vrednost dodeliti takoj, pozneje ali pa sploh ne. Njena vrednost je lahko spremenjena kadarkoli.



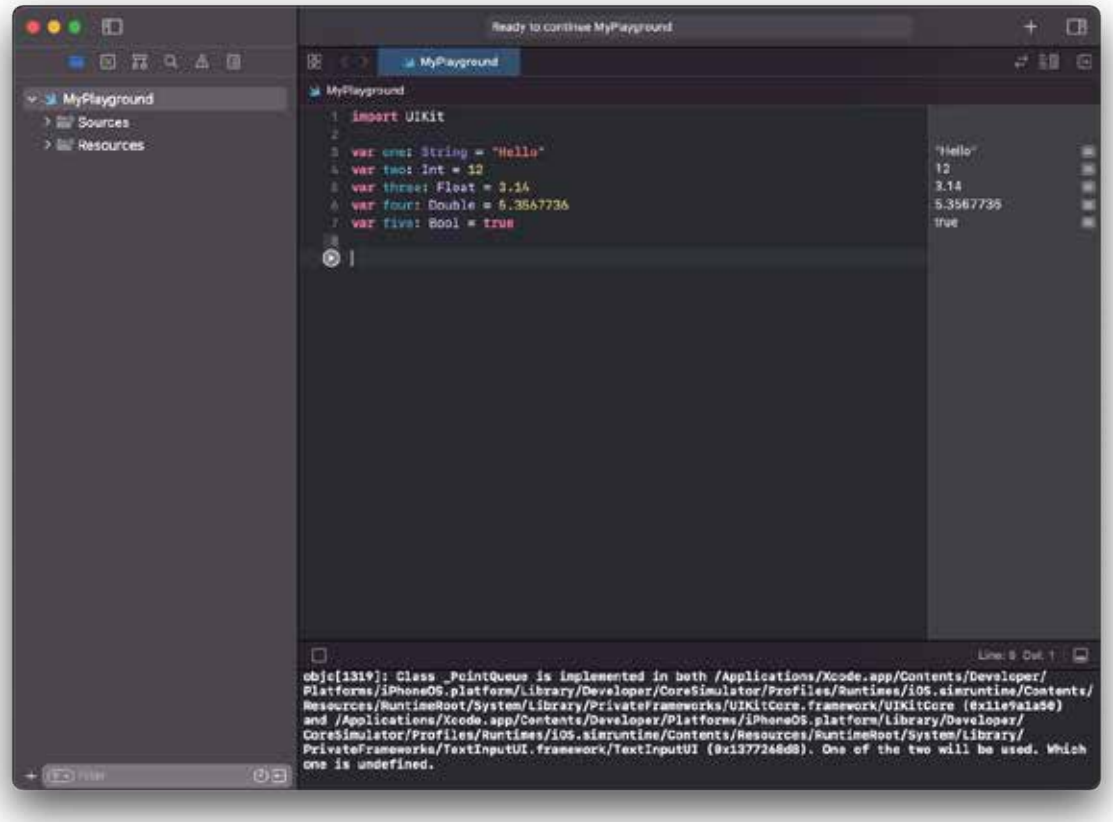
Slika 14: Primerjava podatkovnih tipov var in let.

5.2.2 Glavni podatkovni tipi

Swift ponuja veliko vgrajenih in uporabniško definiranih tipov podatkov. Pri deklaraciji spremenljivk se najpogosteje uporabljajo naslednje:

- **Int:** Uporablja se za cela števila (na primer: 1, 20, –5).
- **String:** Urejena zbirka znakov (na primer: »Hello«).
- **Float:** Predstavlja 32–bitno decimalno število (na primer: 5,43; –3,945).

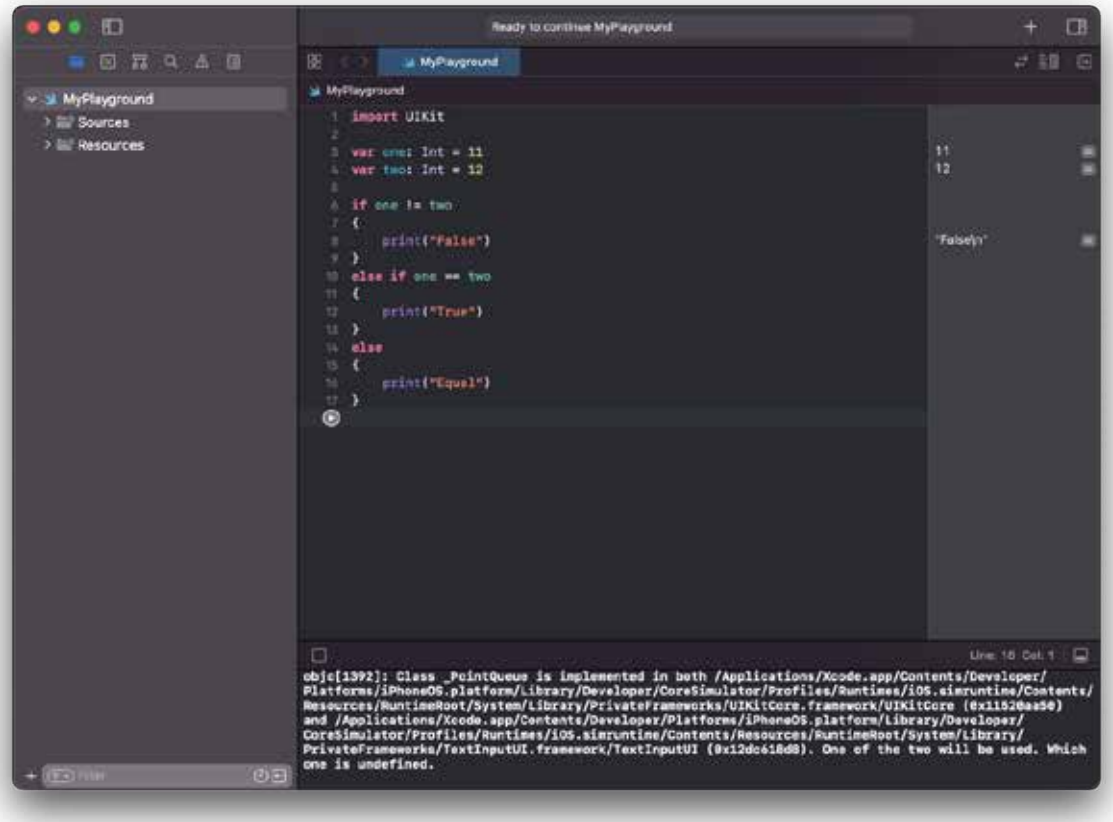
- **Double:** Uporablja se, kadar so decimalna števila izredno velika. Predstavlja 64–bitno število.
- **Bool:** Vsebuje lahko dve vrednosti: true ali false.



Slika 15: Primeri različnih podatkovnih tipov.

5.2.3 If stavek

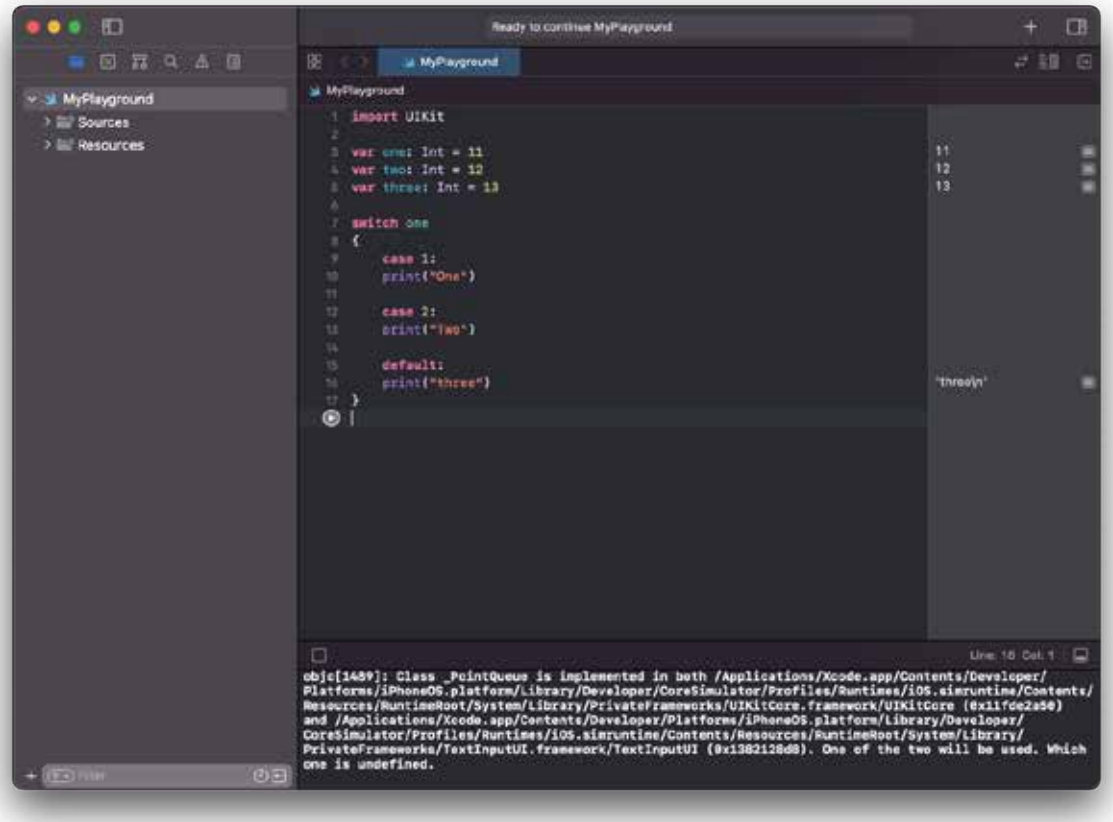
If stavek se v računalniškem programiranju uporablja za zagon dela kode takrat, ko je izpolnjen določen pogoj. Na spodnji sliki je predstavljena sintaksa.



Slika 16: If stavek.

5.2.4 Switch case

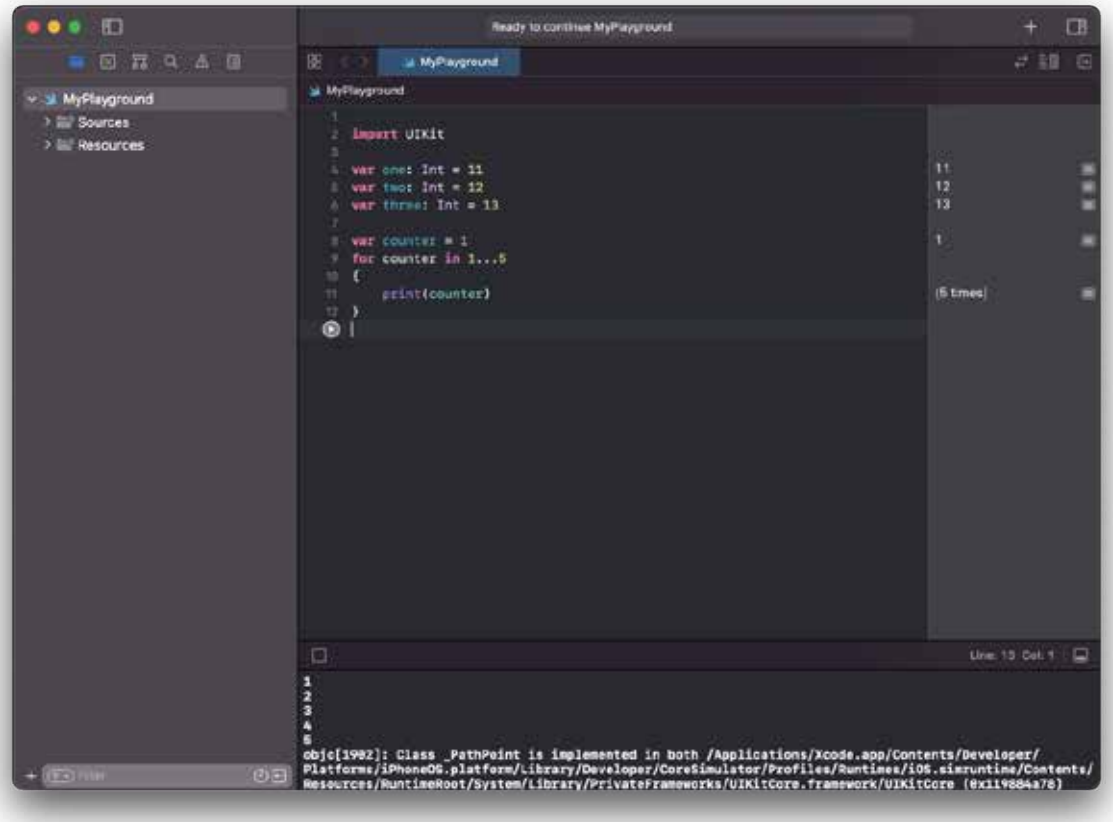
Switch case omogoča, da je izbran eden od več možnih blokov kode za izvajanje. Uporabljen je predvsem tam, kjer je možnih več pogojev. Zanke, kot so If, else if, else, se lahko uporabljajo na enak način, vendar je sintaksa switch case–a bistveno lažja za razumevanje in pisanje.



Slika 17: Switch case.

5.2.5 Zanke

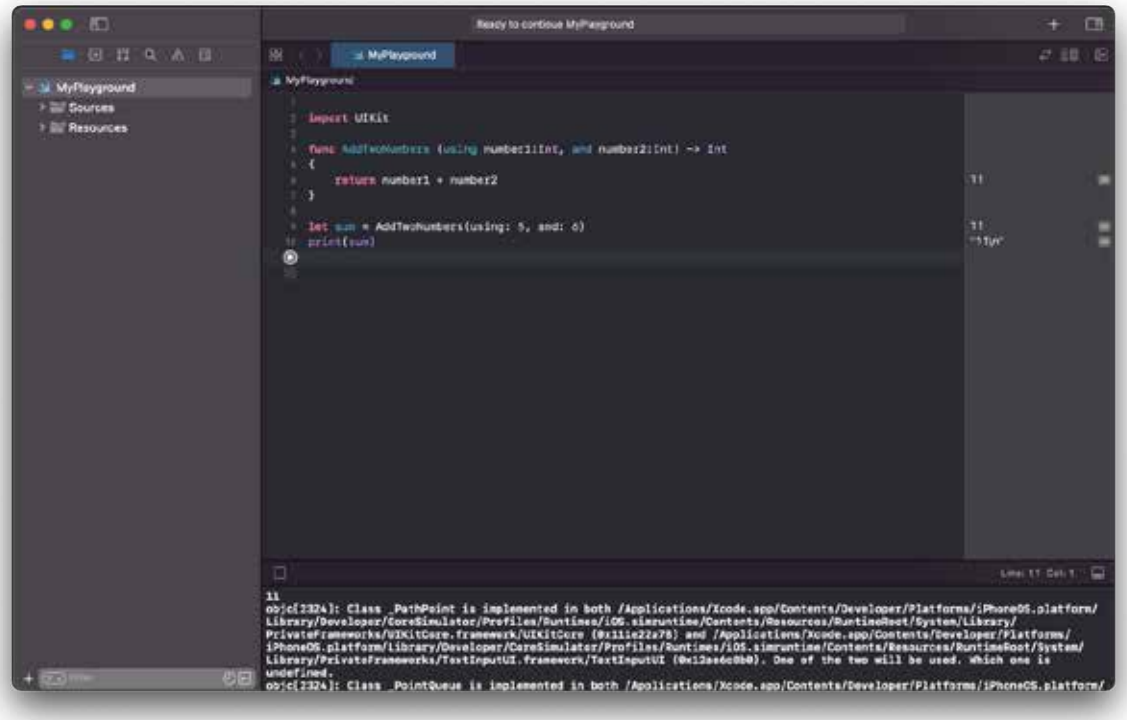
Zanka je niz ukazov, ki se ponavljajo, dokler ni izpolnjen določen pogoj. V zanki je običajno procedura, ki pridobiva in spreminja podatke, sledi pa ji preverjanje pogoja: na primer, ali je neki števec, ki je bil prej inicializiran, dosegel določeno število; če ni, se bo postopek ponavljal, dokler števec ne doseže tega števila. V Swiftu obstaja veliko vrst zank: while, do–while, for ...



Slika 18: For in zanka.

5.2.6 Funkcije

Funkcija je ponovno uporabljiv, urejen kos kode, ki izvaja eno samo povezano dejavnost. Z njo je programu zagotovljena večja modularnost, poleg tega pa se z njimi izognemo ponavljanju kode. Vsak programski jezik ima določene vgrajene funkcije, lahko pa ustvarimo tudi svoje. Funkcije, metode, podrutine, procedure itd. so različna imena zanje v različnih programskih jezikih.



Slika 19: Funkcija.

5.3 Core Motion

Core Motion je framework, ki je vgrajen v Applove operacijske sisteme in omogoča dostop do različnih tipov senzorjev, vgrajenih v iPhone, iPad in Apple Watch, kot so merjenje gibanja (pospeškometri), magnetometer, žiroskop in barometer.

S pomočjo Core Motion lahko razvijalci ustvarijo aplikacije, ki izkoriščajo podatke o gibanju, kot so zaznavanje korakov, merjenje prevožene razdalje, višine in hitrosti, uporabo žiroskopa za nadzor igranja iger in še več.

Nekatere pomembne funkcije, ki jih ponuja Core Motion, so:

- **CMDeviceMotion:** Omogoča dostop do podatkov o pospešku, hitrosti, rotaciji in kalibraciji naprave. Z uporabo te funkcije lahko aplikacije natančno spremljajo gibanje naprave in ga uporabijo za različne namene.
- **CMMotionManager:** Upravlja z vsemi senzorji gibanja naprave. Z njo lahko začnemo in ustavimo spremljanje gibanja, določimo časovni interval posodobitve in spremljamo stanje senzorjev. Uporablja se lahko za merjenje pospeška, rotacije, nagiba in magnetnega polja.
- **CMAccelerometerData:** Zagotavlja podatke o pospešku v treh oseh: x, y in z. Uporablja se lahko za določanje hitrosti, izračun prevožene razdalje ali zaznavanje tresenja naprave.
- **CMGyroData:** Zagotavlja podatke o rotaciji naprave v treh oseh. Uporablja se lahko za nadzor igre z gibanjem ali za določanje orientacije naprave.
- **CMMagnetometerData:** Zagotavlja podatke o magnetnem polju v treh oseh. Uporablja se lahko za določanje smeri naprave ali za zaznavanje magnetnih polj v okolici.

6 Unreal Engine

Unreal Engine je visokozmogljivo razvijalsko okolje za izdelavo videoiger. Razvijalci ga lahko uporabljajo za ustvarjanje različnih vrst iger, kot so akcijske, avanturistične, arkadne, igre za sprostitev in športne igre.

Slika 20: Delovno okolje Unreal Engine.
Vir: https://beforesandafters.com/wp-content/uploads/2021/05/Welcome-to-Unreal-Engine-5-Early-Access-11-16-screenshot.png

Unreal Engine se ponaša z zmogljivo grafično podporo, ki omogoča realistično prikazovanje 3D okolja in objektov v igrah. Vključuje tudi orodja za ustvarjanje in urejanje 3D modelov, tekstur in animacij, kar omogoča razvijalcem, da ustvarijo kompleksne in detajlirane igre.

Ima zmogljive funkcije za upravljanje z igralnimi mehanikami in inteligenco računalniških nasprotnikov. Razvijalci lahko uporabljajo orodja za ustvarjanje pametnih nasprotnikov, ki se lahko prilagodijo igralčevim dejanjem in se obnašajo realistično.

Podpira tudi spletno igranje, tako da lahko razvijalci ustvarijo spletne igre, ki jih igralci igrajo preko spleta. Razvijalci lahko uporabljajo orodja za ustvarjanje in upravljanje spletnih strežnikov za igre, kar omogoča sodelovanje več igralcev.

6.1 Blueprint

Blueprint temelji na vizualnem programiranju, ki ga uporablja Unreal Engine za ustvarjanje interaktivnih iger in aplikacij. Omogoča ustvarjanje in nadzorovanje vedenja igralnih objektov, kot so igralci, predmeti, okolje in drugi elementi.

Blueprinti so sestavljeni iz vozlišč, ki se povezujejo, da ustvarijo določeno vedenje. Obstajajo tri vrste vozlišč:

- **Vhodna vozlišča** sprejemajo vhodne podatke, kot so vhodne spremenljivke, signali dogodkov in vrednosti parametrov.
- **Procesna vozlišča** izvajajo določeno logiko na vhodnih podatkih in vrnejo izhodne podatke.
- **Izhodna vozlišča** vrnejo izhodne podatke, kot so vrednosti spremenljivk, izhodni signali dogodkov ali spremembe stanja igralnega objekta.

Blueprinti so sestavljeni iz več vozlišč, ki so med seboj povezana. Povezave predstavljajo tok podatkov in nadzorujejo izvedbo algoritma.

Blueprinti v Unreal Engine so razdeljeni na tri vrste:

- **Level blueprint.** Blueprint, ki je specifičen za posamezno stopnjo igre. Uporablja se za nadzorovanje interakcije med igralci, predmeti in okoljem.

- **Object blueprint.** Blueprint, ki je specifičen za posamezen objekt v igri. Uporablja se za nadzorovanje vedenja posameznega objekta, kot je premikanje, vrtenje, odkrivanje trkov in interakcije z drugimi objekti.
- **Interface blueprint.** Blueprint, namenjen ustvarjanju vmesnikov, ki se uporabljajo za interakcijo med igralci in objekti.

6.2 VaRest

VaRest (Virtual and RESTful) Subsystem Plugin je vtičnik za Unreal Engine, ki omogoča uporabnikom enostaven dostop do RESTful API–jev (Representational State Transfer) in vsebuje nekaj funkcij za delo z njimi. REST je arhitektura, ki omogoča izmenjavo podatkov med strežniki in klienti, ki uporabljajo HTTP (Hypertext Transfer Protocol).

Vtičnik VaRest omogoča Unreal Engine, da pošilja in prejema podatke preko HTTP protokola, vključno z GET, POST, PUT in DELETE zahtevami. Na ta način lahko razvijalci igre uporabljajo storitve RESTful API–jev za obdelavo in pridobivanje podatkov, kot so uporabniška imena, gesla, lestvice najboljših igralcev in druge informacije, potrebne za njihovo igro.

VaRest vtičnik omogoča tudi enostavno uporabo JSON (JavaScript Object Notation) oblikovanja podatkov, ki je priljubljeno pri RESTful API–jih, in omogoča Unreal Engine, da pretvori JSON podatke v objekte, ki se lahko uporabijo znotraj igre. To je uporabno, ko razvijalci potrebujejo preprost način za pridobivanje in uporabo podatkov, ki jih dobijo od storitve RESTful API–ja.

Vtičnik vključuje tudi funkcije za obdelavo odgovorov RESTful API–jev, kot so uspešnost zahteve, stanje odgovora (HTTP status) in druge. Razvijalci uporabijo te funkcije, da bi zagotovili, da se podatki, ki jih dobijo iz RESTful API–jev, pravilno obdelajo in uporabijo znotraj igre.

7 Python

Python je visokonivojski programski jezik, ki je bil izdan leta 1991 in ga je ustvaril nizozemski programer Guido van Rossum. Danes se Python uporablja za različne namene, vključno z znanstveno analizo podatkov, umetno inteligenco, razvojem spletnih aplikacij, računalniško grafiko in še kaj.

Ena izmed glavnih značilnosti Pythona je njegova berljivost, saj uporablja minimalno število simbolov in ima zelo čisto sintakso. To pomeni, da se koda v Pythonu zlahka bere in razume. Python ima tudi veliko knjižnic, ki vsebujejo mnogo funkcij in modulov, ki se lahko uporabljajo za reševanje različnih nalog.

Slika 21: Python logotip.
Vir: https://www.upf.edu/documents/222910037/261564223/python2.png/7b73d5da-bcc5-72e0-db72-f1e763612931?t=1665491363000

Python je dinamičen jezik, kar pomeni, da se spremenljivke in njihove vrste določijo ob izvajanju programa. Poleg tega se Python izvaja s pomočjo interpretatorja, kar pomeni, da kode ni treba predhodno prevajati, saj se to dogaja v času izvajanja.

Python podpira različne vrste podatkovnih tipov, vključno z nizi znakov, števili, seznami in množicami. Python ima tudi veliko vgrajenih funkcij in modulov, ki lahko pomagajo pri obdelavi in manipulaciji podatkov.

Python se lahko uporablja na različnih platformah, kot so Windows, Linux in macOS. Poleg tega se lahko integrira z drugimi programskimi jeziki in platformami.

Za razvoj v Pythonu je na voljo veliko orodij, vključno z integriranim razvojnim okoljem (IDE) PyCharm, notranjim razvojnim okoljem (IDE) IDLE in orodjem za urejanje besedila Sublime Text.

V Pythonu lahko razvijamo različne vrste aplikacij, vključno z desktop aplikacijami, spletnimi aplikacijami, mobilnimi aplikacijami in igrami. Poleg tega se Python uporablja za različne druge namene, vključno z analizo podatkov, strojnim učenjem, umetno inteligenco, razvojem interneta stvari in podobno.

7.1 Flask

Flask je zmogljiva mikroknjižnica za razvoj spletnih aplikacij v Pythonu. Flask je ustvaril Armin Ronacher in je priljubljen med razvijalci, ki iščejo preprost in fleksibilen način za gradnjo spletnih aplikacij.

Slika 22: Flask logotip.
*Vir: https://miro.medium.com/max/640/1*XzIRJGujfqAiOV2EIQgR_Q.png*

Glavne značilnosti Flaska so:

- **Minimalizem:** Osredotoča se na osnovne funkcije za gradnjo spletnih aplikacij. Ima zelo majhen obseg in je zato lažji za učenje in uporabo.
- **Prilagodljivost**: Zasnovan je tako, da se lahko prilagodi za uporabo z drugimi orodji in knjižnicami. Ne predpisuje nobenih specifičnih praks ali predpostavk o vaši aplikaciji.
- **Podpora za razširitve:** Flask ima številne razširitve, ki olajšajo delo z različnimi nalogami, kot so avtentikacija uporabnikov, oblikovanje obrazcev, delo z bazami podatkov in podobno.
- **Vgrajen razhroščevalnik:** Flask vsebuje vgrajen razhroščevalnik, ki pomaga pri odkrivanju in odpravljanju napak v aplikaciji.
- **Spletni strežnik:** Flask vsebuje vgrajen spletni strežnik, ki ga je enostavno zagnati in uporabiti.

Flask vključuje številne funkcije, ki pomagajo pri gradnji spletnih aplikacij, kot so:

- **Upravljanje zahtevkov:** Flask zagotavlja enostaven način za obdelavo in upravljanje zahtevkov, vključno z upravljanjem URL–jev, oblikovanjem in obdelavo obrazcev.

- **Podpora za predloge:** Flask vključuje podporo za predloge, ki omogoča, da se aplikacija dinamično prilagodi glede na uporabnikove zahteve.
- **Baze podatkov:** Flask podpira različne vrste baz podatkov, vključno z relacijskimi in nestrelacijskimi bazami podatkov.
- **Spletno varnost:** Flask vključuje številne funkcije, ki pomagajo pri izboljšanju varnosti aplikacije, vključno z upravljanjem uporabnikov, šifriranjem podatkov in zaščito pred vdori.
- **Integracija z drugimi knjižnicami:** Flask se lahko integrira z drugimi Python knjižnicami, kot so NumPy, Pandas in Scikit–learn, kar omogoča lažje delo z različnimi nalogami, kot so analiza podatkov in strojno učenje.

Flask je odprtokodni projekt in je na voljo na platformi GitHub.

8 Golf z uporabo pametne ure

8.1 Idejna zasnova

Igro smo si zamislili kot simulator golfa, ki omogoča igralcem, da se preizkusijo na različnih igriščih z različnimi težavnostmi. Glavni poudarek igre bi bil na simulaciji fizičnih lastnosti žogice, kot so hitrost, smer in razdalja, ki so odvisne od igralčevega udarca in izbire palice.

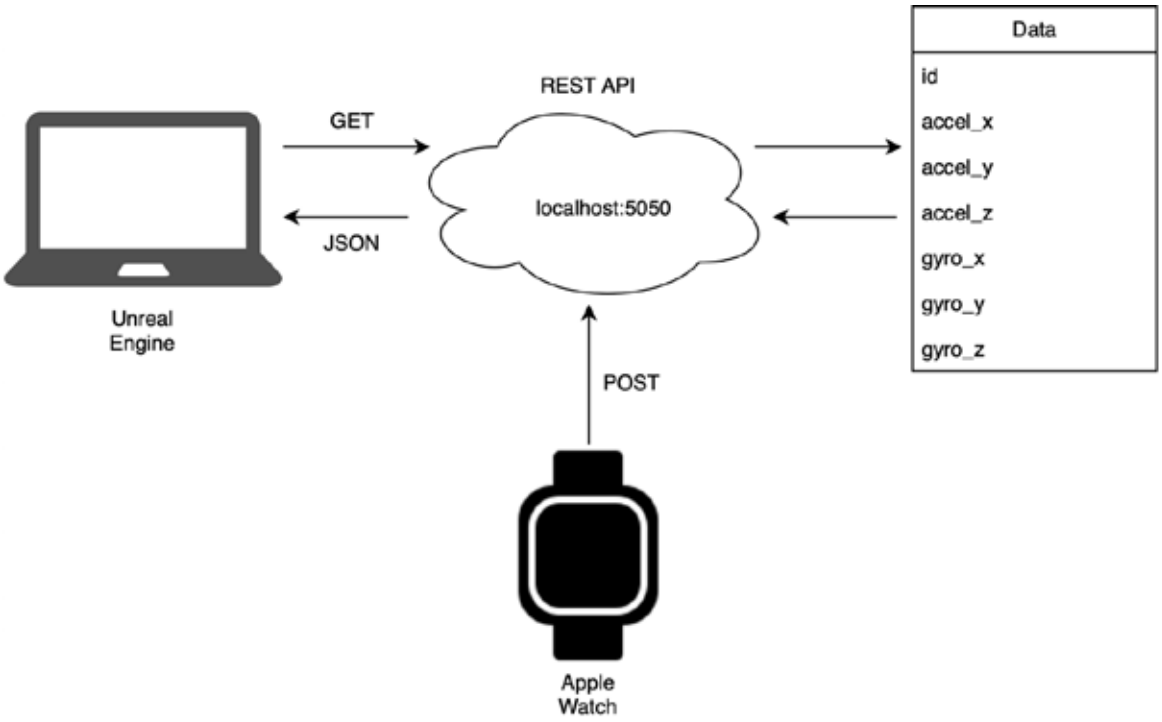
Za merjenje moči udarca bi uporabili pametno uro Apple Watch, ki bi spremljala gibanje roke igralca in poslala podatke preko brezžične povezave na računalnik, na katerem se bo izvajala igra. Podatki bi se nato uporabili za izračun gibanja žogice na igrišču.

8.2 Tehnična zasnova

Na strežniku, ki deluje na naslovu localhost:5050, se nahaja podatkovna baza, v katero se shranjujejo podatki o udarcih, ki jih meri Apple Watch. Podatki se prenašajo preko zahtev HTTP POST, kar pomeni, da se pošiljajo na strežnik za shranjevanje.

Računalnik, na katerem teče Unreal Engine, prejema podatke s strežnika z zahtevo HTTP GET. Ta zahteva se uporablja za pridobivanje podatkov s strežnika, kar v tem primeru pomeni prenos podatkov o prejšnjih udarcih, ki so bili shranjeni v podatkovno bazo.

Podatki se v obliki JSON (JavaScript Object Notation) pošiljajo med napravami. Ta format podatkov je lahek in enostaven za uporabo, zato je zelo primeren za prenos podatkov med različnimi napravami in programskimi okolji.



Slika 23: Shematski prikaz komunikacije med napravami.

8.3 Komunikacijski protokoli

8.3.1 HTTP POST

HTTP POST je protokol, ki se uporablja za pošiljanje zahtevkov na strežnik, da bi ustvarili nove vire. POST zahteva vključuje glavo in telo, ki vsebuje podatke, ki jih želimo poslati na strežnik. Poleg tega se uporablja tudi za pošiljanje podatkov z uporabo obrazcev HTML, kjer uporabnik vpiše podatke v obrazec, ta pa se pošlje s POST zahtevo.

Skupaj z glavo (ki vključuje informacije, kot so URL cilja, tip vsebine in velikost telesa zahtevka) se pošlje tudi telo zahtevka (v tem primeru JSON), ki vsebuje dejanske podatke, ki jih je treba poslati na strežnik.

Nekatere aplikacije API lahko zahtevajo dodatne avtorizacijske podatke, kot so žetoni za avtentikacijo, da se preveri, ali je oseba, ki pošilja POST

zahtevek, pravilen uporabnik, ki ima dovoljenje za ustvarjanje novih virov. V tem primeru bi bili ti podatki dodani v glavo zahtevka.

V našem primeru uporabljamo POST zahtevek za pošiljanje podatkov (podatek o pospešku in podatek o rotaciji) iz pametne ure na strežnik. Ko pošljemo POST zahtevek na strežnik, se ta zahteva obdeluje na strežniški strani, nato se ustvari nov vir z uporabo podatkov iz telesa zahtevka. To se pokaže v obliki novih vnosov v baze podatkov.

Primer HTTP POST zahtevka, ki ga uporabljamo:

```
POST / HTTP/1.1 200
Host: localhost:5050
Content–Type: application/json
{
»accel_x«: »0.017216«,
»accel_y«: »0.0197484«,
»accel_z«: »–0.0328017«,
»gyro_x«: »1.95837«,
»gyro_y«: »1.13106«,
»gyro_z«: »0.029933«
}
```

To je HTTP POST zahtevek, ki se pošlje na localhost in vsebuje JSON podatke v telesu zahteve. Vsebuje štiri podatke o senzorjih: »accel« predstavlja pospešek, ki je zabeležen v enotah g (vrnjena 1 pomeni vrednost gravitacijskega pospeška (g), to je 9,81 m/s^2), »gyro« pa predstavlja hitrost vrtenja, ki je zabeležena v enotah °/s. Oznake »x«, »y« in »z« predstavljajo smeri. Odgovor strežnika bo statusna koda 200, kar pomeni, da je zahteva uspešno obdelana.

8.3.2 HTTP GET

HTTP GET je eden od osnovnih zahtevkov, ki jih uporabljajo spletni odjemalci (spletni brskalniki, mobilne aplikacije itd.) za pridobivanje podatkov s spletnega strežnika. Zahtevek HTTP GET odjemalcu omogoča pridobivanje vsebine spletnih strani, slik, videov, datotek ali drugih virov, ki so dostopni preko interneta.
GET zahtevek vsebuje tri pomembne sestavine:

- **URL (Uniform Resource Locator) naslov:** Naslov, ki se zahteva. Primer URL–ja: *https://www.example.com/page.html*.
- **HTTP metoda:** Specifikacija, katero metodo uporablja odjemalec. Zahtevan je pri vsakem HTTP zahtevku. GET metoda se uporablja, ko želimo pridobiti podatke s spleta.
- **Header polja:** Specifikacija dodatnih informacij za zahtevek. Primer: tip dokumenta, ki ga odjemalec pričakuje od strežnika.

HTTP GET zahtevek pošilja podatke neposredno preko URL–ja. V primeru, da URL vsebuje parametre, se ti določijo v URL–ju. Primer: *https://www.example.com/page.html?param1=value1&param2=value2*.

V našem primeru gre za HTTP GET zahtevek, ki se pošilja na lokalni strežnik z naslovom »localhost« na vrata 5050. Zahtevek uporablja HTTP protokol verzije 1.1 in vsebuje tudi HTTP statusno kodo 200, ki pomeni, da je zahteva uspešna. Vsebuje tudi »Content–Type«, ki specificira, da se v telesu zahtevka nahaja JSON podatkovna oblika. V zahtevku uporabnik (Unreal Engine) povprašuje po podatku o pospeških in rotacijah. Te podatke bo Unreal Engine potreboval za izvajanje udarcev.

Primer HTTP GET zahtevka, ki ga uporabljamo:

```
GET / HTTP/1.1 200
Host: localhost:5050
Content–Type: application/json
```

Strežnik po prejemu HTTP GET zahtevka preveri, ali obstaja vsebina, ki jo je odjemalec zahteval, in v primeru obstoja vrne vsebino v obliki HTTP odgovora. Vsebina HTTP odgovora vsebuje podatke o pospeških in rotacijah.

Primer odgovora HTTP GET zahtevka:

```
{
»accel_x«: »0.028693«,
»accel_y«: »0.019309«,
»accel_z«: »–0.040988«,
»gyro_x«: »1.09823«,
»gyro_y«: »1.3478«,
»gyro_z«: »0.028922«
}
```

8.3.3 TCP/IP

TCP/IP je protokolni sklad, ki se uporablja za prenos podatkov preko omrežja. Ime TCP/IP izhaja iz dveh protokolov, ki tvorita sklad: Transmission Control Protocol (TCP) in Internet Protocol (IP).

Internet Protocol (IP) je splošen protokol, ki se uporablja za prenos podatkov v omrežju. IP zagotavlja podatkovni tok med različnimi omrežnimi napravami in določa, kako se podatki pakirajo, naslovijo, prenašajo in dostavijo. IP uporablja edinstvena naslovna mesta, znana kot IP naslovi, za identifikacijo naprav v omrežju. IP se uporablja na nižji ravni protokolnega skladovja kot TCP.

Transmission Control Protocol (TCP) zagotavlja zanesljiv prenos podatkov med aplikacijami, ki se izvajajo na različnih napravah v omrežju. TCP je

zasnovan tako, da zagotavlja, da se vsi podatki prenašajo brez napak in v pravilnem vrstnem redu. Poleg tega TCP zagotavlja, da se podatki prenašajo po najboljši poti med napravami v omrežju. TCP se uporablja na višji ravni protokolnega skladovja kot IP.

TCP in IP zagotavljata, da se podatki prenašajo brez napak in v pravilnem vrstnem redu med različnimi aplikacijami, ki se izvajajo na različnih napravah v omrežju. TCP/IP protokolni sklad se uporablja kot osnova za omrežne komunikacije na svetovnem spletu in v številnih drugih omrežjih.

8.4 Aplikacija na Apple Watch

Najpomembnejšo vlogo pri igri ima Apple Watch. Igralcu služi kot naprava za zajemanje podatkov o gibanju zapestja med udarci s palico in o pospešku, kar omogoča merjenje moči udarcev.

8.4.1 Začetni zaslon

Uporabniški vmesnik te aplikacije je preprost in minimalističen. Na začetnem zaslonu se nahajata dva gumba: za začetek in pomoč. Ob pritisku gumba za začetek se vzpostavi povezava pametne ure s strežnikom in nanj se začnejo pošiljati podatki. Če uporabnik pritisne gumb pomoč, se izpiše nekaj osnovnih podatkov o aplikaciji.

Slika 24: Začetni zaslon aplikacije na Apple Watch.

8.4.2 Zaslon pri igri

Na vrhu zaslona je besedilo, ki opisuje, kaj se trenutno dogaja (»Pošiljanje podatkov na strežnik ...«). Pod tem besedilom se nahaja prazen prostor. Na sredini zaslona je gumb z besedilom »Nazaj«, ki je namenjen vrnitvi na prejšnji zaslon.

Slika 25: Zaslon pri igranju golfa na Apple Watch.

Ta aplikacija nima veliko funkcij, ki bi jih igralec lahko uporabljal, saj je namenjena predvsem zbiranju podatkov iz senzorjev na Apple Watchu in pošiljanju teh podatkov na strežnik. Zato je uporabniški vmesnik preprost in ima le nekaj elementov. Celoten izgled je minimalističen in enostaven za uporabo, tako da uporabniku ni treba razmišljati o uporabi aplikacije, temveč se lahko osredotoči na igro golfa.

Za zajemanje podatkov o gibanju se uporabljajo funkcije Core Motion, ki jih vsebuje Apple Watch. S pomočjo teh funkcij se zbirajo podatki o pospeških in hitrostih gibanja zapestja med udarci. Igra uporablja te podatke za merjenje razdalje udarcev.

Ko igralec udari žogo, se podatki o gibanju zapestja in pospeških prenesejo na strežnik prek funkcije postMethod(). Ta funkcija pretvori podatke v JSON format, jih pošlje na strežnik prek HTTP POST zahtevka in nato prejme odgovor s strežnika, ki vsebuje podatke o posodobljenem stanju igre. Pri igri se uporablja tudi funkcija postMethod(), in sicer za testiranje, saj pošilja velike količine naključnih podatkov na strežnik.

8.4.3 Zaslon za pomoč

Na zaslonu za pomoč se nahajajo osnovni podatki o aplikaciji.

Slika 26: Zaslon za pomoč na Apple Watch.

8.5 Aplikacija za računalnik

Aplikacija na računalniku, na katerem teče Unreal Engine, služi kot igralni zaslon za igro. Igralcu omogoča igranje na računalniku, kjer se prikazujejo 3D vizualizacije igralnega okolja in drugih elementov igre.

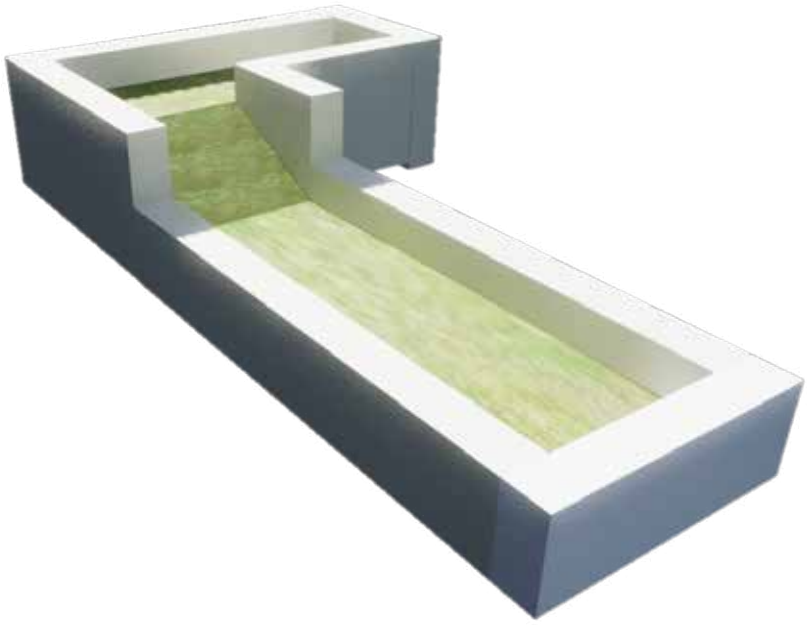
Igra je simulacija golfa, kjer se igralec znajde v prvoosebnem pogledu, kar pomeni, da so igralčeve oči pripete na žogo, kamera pa ji ves čas sledi. Igra ima tri različne stopnje, pri čemer se težavnost z vsako naslednjo povečuje.

Slika 27: Prvoosebni pogled na igrišče.

Ko igralec zažene igro, se najprej prikaže začetni meni, na katerem ima dve možnosti: začeti ali zapustiti igro. Če se odloči za začetek igre, se naloži prva stopnja, ki jo mora uspešno dokončati, da nadaljuje na naslednjo.

Slika 28: Prva stopnja igre.

Vsaka stopnja se začne s postavitvijo žoge na začetno mesto, igralec pa mora žogo potisniti v luknjo, ki je postavljena na različnih lokacijah.



Slika 29: Druga stopnja igre.

Ko igralec uspešno konča vse tri stopnje, ga igra preusmeri na začetni meni, kjer ima ponovno možnost začeti ali zapustiti igro. To mu omogoča, da lahko igro ponovno začne in ponovi stopnje ter svoj dosežek izboljša.

Glavno orodje pri igri je Apple Watch, od katerega Unreal Engine pridobi vse potrebne podatke (pospeški, vrtenje). Na podlagi teh izračuna, kako hitro se bo premaknila žoga in v katero smer. Da smo omogočili medsebojno komunikacijo med Unreal Engine in Apple Watch, smo uporabili Va Rest Subsystem v Unreal Engine–u. Povezan je s funkcijo, v kateri je shranjen URL strežnika, s katerim komuniciramo. URL strežnika je naslov, ki nam omogoča dostop do strežnika preko omrežja. Ko želimo prejeti podatke s strežnika, pošljemo HTTP GET zahtevek na določen URL naslov. Strežnik nato vrne podatke v obliki JSON datoteke, ki jih Unreal Engine prebere in

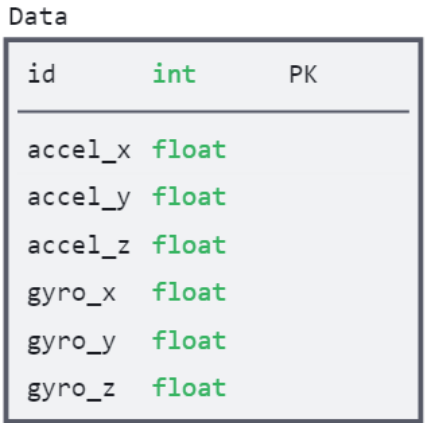
uporabi za nadaljnjo obdelavo. V našem primeru na ta način prejmemo podatke o položaju in hitrosti premika ure. Te podatke nato uporabimo za premikanje žoge.

Pomembno je zagotoviti, da je strežnik, s katerim se sistem povezuje, vedno dosegljiv. Če strežnik ni dosegljiv, ne bomo prejeli podatkov in igra ne bo delovala pravilno. Da bi zagotovili dosegljivost strežnika, smo morali poskrbeti za ustrezno omrežno povezljivost, varnost in nadzor.

8.6 Strežnik

Strežnik uporabljamo izključno za REST API in podatkovno bazo. Ker nihče od nas ni bil vešč programskega jezika, kot je JavaScript, smo se odločili, da bomo uporabili Python skupaj s knjižnico Flask. Ta knjižnica je oblikovana tako, da podpira RESTful oblikovanje, ki omogoča razvijalcem, da oblikujejo REST API v skladu s standardi REST arhitekture. Načrtovanje strežnika ni bilo prezahtevno, saj smo uporabljali samo POST in GET zahteve. Flask sicer ni bila naša prvotna izbira; najprej smo načrtovali strežnik s knjižnico Socket, a smo naleteli na težave pri povezavi z Apple Watch, saj tega ni podpirala. Nato smo testirali uporabo http.server knjižnice, vendar smo naleteli na težave pri preverjanju zahtev. Nazadnje smo se odločili za Flask. Pri dosedanjem testiranju nismo našli veliko težav, razen občasnih prekinitev delovanja zaradi prevelikega števila zahtev.

Na strežniku se nahaja podatkovna baza, ki je sestavljena iz tabele »Data«, v kateri se shranjuje »id« podatka (z uporabo UPDATE SQL stavka, namenjenega spreminjanju vrednosti), »gyro« podatki (vrednosti vrtenja ure) in »accel« podatki (hitrosti premikanja ure). Do podatkovne baze ima dostop samo strežnik, ki pregleduje prejete zahteve. Geslo do baze je šifrirano z Base64 algoritmom, tako da je zagotovljena varna povezava in ni mogoče nepooblaščeno spreminjanje podatkov.



Slika 30: Tabela v podatkovni bazi.

Strežnik glede na URL sproži fukcijo, ki preverja vrsto zahteve. V primeru POST zahteve strežnik iz JSON stringa izlušči pomebne podatke (gyro, accel), da jih lahko zapiše v tabelo »Data«. Z vsakim GET zahtevkom pa strežnik iz podatkovne baze (SQL poizvedba) prejme podatke, jih zapakira v JSON string in pošlje naprej. Povezava je šifrirana z API ključem, ki je šifriran z Base64 algoritmom. Primer: če bi poznali URL pot, ki ne potrebuje API ključa, bi samo poslali POST zahtevo in lahko bi spreminjali podatke, ne glede na to, da ne poznamo gesla podatkovne baze, zato ga še dodatno preverjamo z API ključem, ki zagotovi, da lahko na strežnik pošilja oziroma dostopa samo uporabnik z znanim API ključem.

8.7 Uporabnost aplikacije

Aplikacija, ki smo jo razvili, je igra, namenjena simulaciji golf igre s pomočjo pametne ure in računalnika. Pametna ura služi kot senzor za merjenje udarcev, računalnik pa za prikaz golf igrišča. Igra igralcu omogoča, da lahko preizkusi golf; tistim, ki ga že igrajo, pa omogoča alternativo h klasični golf igri.

Izboljšave za prihodnje bi lahko vključevale dodajanje novih funkcij, kot je na primer analiza udarcev, ki bi igralcem pomagala pri izboljševanju tehnike, in integracija z drugimi aplikacijami, kot so aplikacije za družabna omrežja, ki bi omogočale deljenje dosežkov in izzivov med igralci. Prav tako bi lahko razvili večjo prilagodljivost aplikacije, da bi jo lahko prilagodili različnim vrstam igralcev in različnim vrstam igrišč. Lahko bi omogočili shranjevanje podatkov o posameznih igrah in ocenjevanje napredka igralca.

V prihodnje bi lahko razvili tudi večplatformno aplikacijo, ki bi bila na voljo za različne mobilne operacijske sisteme, kar bi povečalo dostopnost aplikacije za večje število uporabnikov. Poleg tega bi lahko aplikacijo nadgradili z umetno inteligenco, ki bi lahko pomagala pri sprejemanju boljših odločitev med igro.

Vse te izboljšave bi povečale uporabnost aplikacije za golf igralce in jim pomagale pri izboljševanju njihove igre in pripravi na tekme.

9 Rezultati

Naša raziskovalna naloga se je osredotočala na razvoj aplikacije, ki bi omogočala uporabo Apple Watcha v golf igri. V teoretičnem delu smo predstavili naprave, na katerih bi tekla igra, vključno s pametno uro in računalniškim vmesnikom. Prav tako smo predstavili aplikacije, ki so bile uporabljene pri raziskovanju, in se osredotočili na zgradbo in delovanje igre.

V praktičnem delu smo uspešno razvili golf igro, ki uporablja pametno uro in računalnik. Uporabili smo različne senzorje na Apple Watchu in od njih pridobili podatke, kot sta pospešek in smer. Te podatke smo nato uporabili za simulacijo golf udarcev.

Povezali smo tudi različne naprave in strežnike, da bi zagotovili, da se igra izvaja brezhibno in brez zakasnitve. Občutek pravega golfa smo dosegli tako, da igra sledi igralčevim zamahom.

Celotno funkcionalnost smo zapakirali v uporabniško izkušnjo, ki je intuitivna in enostavna za uporabo. Uporabniki lahko z uporabo aplikacije brezplačno preizkusijo golf igro in se odločijo, ali bi želeli nadaljevati s tovrstno tehnologijo.

Ocenili smo, da bi lahko aplikacija pritegnila širši krog ljudi, ki bi želeli preizkusiti golf igro, ne da bi se morali odločiti za nakup opreme ali najem igrišča.

Na podlagi intervjuja smo ugotovili, da bi aplikacija lahko pripomogla k boljši igri. Igralec golfa, ki smo ga intervjuvali, meni, da bi bilo izvajanje simulacij udarcev z uporabo pametne ure zelo koristno, še posebej pri testiranju različnih površin. Prav tako meni, da bi aplikacija lahko bila koristna za začetnike, saj bi jim pomagala pri učenju igre in pravilni uporabi različnih palic. Pri profesionalnih igralcih bi aplikacija lahko postala konkurenca

simulatorjem, če bi imele vse funkcije, ki jih ti ponujajo. Želi si še, da bi aplikacija omogočala prilagajanje igre, spremljanje napredka in dostop do baze podatkov o igralnih progah in luknjah. Dodatno bi si želel merjenja razdalj do ovir in luknje, kar bi mu pomagalo pri odločitvah o izbiri palice.

10 Ovrednotenje hipotez

1. HIPOTEZA: Zaradi zaprtega Apple sistema se bomo srečali s težavami.
To hipotezo lahko delno potrdimo. Zaprtost Apple sistema je sicer predstavljala nekaj omejitev, ker smo morali pri razvoju aplikacije za uro uporabljati izključno Apple naprave. Vendar pa smo kljub temu lahko razvili funkcionalno igro, ki izkorišča vse prednosti Apple Watcha in drugih Apple naprav.

2. HIPOTEZA: Potrebno bo uporabiti game engine.
Ta hipoteza se je izkazala za resnično. Uporabili smo Unreal Engine, ki nam je omogočil, da smo hitro in učinkovito ustvarili igro z bogatimi vizualnimi efekti.

3. HIPOTEZA: Potrebno bo dopolniti komunikacijske protokole.
Te hipoteze ne moremo potrditi. Naša aplikacija je namreč temeljila na obstoječih protokolih, kot sta HTTP GET in TCP/IP, ki so se izkazali za dovolj učinkovite za prenos podatkov med napravami. Zato lahko zaključimo, da dopolnjevanje komunikacijskih protokolov ni bilo potrebno.

4. HIPOTEZA: Projekt presega srednješolsko znanje.
Ta hipoteza se je izkazala za delno resnično. Razvoj igre je zahteval nekoliko naprednejše programersko znanje, vendar pa smo s pomočjo mentorja in dodatnega izobraževanja lahko razvili funkcionalno igro.

5. HIPOTEZA: Igra bo izboljšala pripravljenost golfistov na tekme.
Ta hipoteza ni bila preverjena s praktičnimi testi, saj nam časovni okvir raziskave ni omogočal izvedbe raziskave na širši skupini golfistov. Vendar pa smo izvedli intervju, v okviru katerega smo predstavili našo aplikacijo. Odziv je bil dober in igralec golfa pravi, da ima aplikacija potencial, tako za začetniško kot tudi za profesionalno rabo.

6. HIPOTEZA: Igra bo pritegnila potencialne nove igralce golfa.

Ta hipoteza ni bila preverjena s praktičnimi testi, saj nam časovni okvir ni omogočal raziskave na širši skupini potencialnih golfistov. Vendar pa bi bilo to zanimivo preveriti v nadaljnjih raziskavah.

7. HIPOTEZA: Stroški projekta ne bodo presegli 200 €.

Ta hipoteza se je izkazala za resnično. Pri raziskovanju smo potrebovali dražjo opremo, kot sta MacBook in Apple Watch, vendar smo jo že imeli, zato zaradi projekta nismo imeli dodatnih stroškov. Prav tako smo uporabljali brezplačne programe in orodja.

11 Zaključek

V okviru te raziskovalne naloge smo se lotili izdelave golf igre s pomočjo pametne ure Apple Watch in računalnika. Cilj je bil uporabnikom omogočiti alternativo h klasični golf igri, ki bi jim lahko pomagala izboljšati njihovo igro, obenem pa prihranila stroške za opremo in najem igrišča.

Pri raziskovanju smo se osredotočili na predstavitev naprav, na katerih bo tekla igra, in aplikacij, ki so bile uporabljene pri raziskovanju. Predstavili smo tudi zgradbo in delovanje igre ter raziskali, na kakšne načine se lahko uporablja igra.

V praktičnem delu smo uspešno naredili golf igro s pomočjo pametne ure in računalnika, uporabili smo različne senzorje na Apple Watch in od njih pridobili podatke. Povezali smo tudi različne naprave in strežnike ter nudili občutek prave golf igre, ki smo jo zapakirali v odlično uporabniško izkušnjo.

Naša raziskava je pokazala, da smo kljub zaprtemu Apple sistemu uspeli izdelati igro. Potrebno je bilo uporabiti game engine. Komunikacijskih protokolov ni bilo treba dopolnjevati. Projekt je presegel srednješolsko znanje, vendar smo se s trudom in raziskovanjem uspešno spopadli s izzivi.

Upamo, da bo glede na končne rezultate igra izboljšala pripravljenost golfistov na tekme ter pritegnila tudi nove igralce. Celotna raziskava in izdelava igre sta bili zelo zanimivi in poučni ter sta nam omogočili pridobivanje novih izkušenj na področju programiranja, senzorike in komunikacij.

12 Viri in literatura

Buttfield-Addison, P., Manning, J., & Nugent, T. (2017). *Learning Swift: Building Apps for macOS, iOS, and Beyond, 2nd Edition*. California: O'Reilly Media.

Altexsoft. (13. september 2021). *Blog*. Pridobljeno 2. novembra 2021 iz Altexsoft: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/

Bohon, C. (25. oktober 2021). *Developer*. Pridobljeno 2. novembra 2021 iz TechRepublic: https://www.techrepublic.com/article/apples-swift-programming-language-the-smart-persons-guide/

Neznani avtor. (brez datuma). *Apple Watch*. Pridobljeno 5. januarja 2023 iz Wikipedia: https://en.wikipedia.org/wiki/Apple_Watch

Neznani avtor. (brez datuma). *Swift (programming language)*. Pridobljeno 5. januarja 2023 iz Wikipedia: https://en.wikipedia.org/wiki/Swift_(programming_language)

Neznani avtor. (brez datuma). *Unreal Engine*. Pridobljeno 5. januarja 2023 iz Wikipedia: https://en.wikipedia.org/wiki/Unreal_Engine

Schaffer, E. (27. oktober 2021). *Blog*. Pridobljeno 2. novembra 2021 iz Educative: https://www.educative.io/blog/swift-programming

Shepherd, A. (24. marec 2021). *Development*. Pridobljeno 2. novembra 2021 iz ITPro: https://www.itpro.co.uk/development/34417/what-is-the-swift-programming-language-and-why-should-i-learn-it

Neznani avtor. (brez datuma). *Python (programming language)*. Pridobljeno 21. februar 2023 iz Wikipedia: https://en.wikipedia.org/wiki/Python_(programming_language)

Neznani avtor. (brez datuma). *Xcode*. Pridobljeno 1. marca 2023 iz Wikipeda: https://en.wikipedia.org/wiki/Xcode

Neznani avtor. (brez datuma). *User's Guide*. Pridobljeno 1. marca 2023 iz Flask: https://flask.palletsprojects.com/en/2.2.x/

Neznani avtor. (brez datuma). *Unreal Engine 4 Documentation*. Pridobljeno 21. februarja 2023 iz Unreal Engine: https://docs.unrealengine.com/4.26/en-US/

Krüger, A. (2017). *Recognizing and classifying a golf swing using accelerometer in a Smartwatch*. Pridobljeno 1. marca 2023 iz DiVA portal: https://www.diva-portal.org/smash/get/diva2:1114198/FULLTEXT01.pdf

13 Priloge

13.1 Intervju

Kakšna je vaša izkušnja z igranjem golfa in ali že uporabljate kakšne tehnološke pripomočke pri igranju?
Že nekaj let ljubiteljsko igram golf. Pri igranju ne uporabljam posebnih tehnoloških pripomočkov.

Ali bi bilo koristno, če bi lahko s pomočjo pametne ure Apple Watch izvajali simulacije udarcev na različnih površinah? Kako bi to lahko vplivalo na vaše nadaljnje treninge?
Menim, da bi bilo takšno izvajanje simulacij zelo koristno, tako za amaterske kot profesionalne igralce golfa. Verjetno bi nam pomagalo izboljšati tehniko in lahko bi testirali različne terene (trda podlaga, pesek, zelena površina), ne da bi stopili iz naše cone udobja.

Kako bi se po vašem mnenju Apple Watch lahko uporabljala za izboljšanje golfa za začetnike?
Apple Watch bi lahko začetnikom pomagal pri učenju igre in pri pravilni uporabi različnih palic. Poleg tega bi lahko aplikacija ponudila nasvete o tem, kako izboljšati svojo igro in kje najbolj pogosto delajo napake.

Kako bi se po vašem mnenju takšna simualcija obnesla pri profesionalnih igralcih golfa?
Kolikor vem, golf simulatorji potrebujejo veliko prostora in za večino niso cenovno dostopni. Tako da — če bi igra omogočala vse, kar omogoča golf simulator, bi lahko ta aplikacija postala dovolj dobra konkurenca tem simulatorjev, ampak se še vseeno pojavi problem občutka, ki ga ima igralec, ko v roki drži pravo palico. V imenu profesionalnih igralcev golfa ne morem reči, kako bi doživeli to izkušnjo.

Kakšne funkcije bi si želeli v aplikaciji za Apple Watch, ki bi jo uporabljali med igro golfa?

Kar se tiče funkcij, bi si želel možnosti za prilagajanje svoje igre, spremljanje in beleženje napredka v realnem času ter dostop do bogate baze podatkov o različnih igralnih progah in luknjah. Poleg tega bi lahko v aplikaciji uporabljali tudi funkcije za fizično pripravljenost, ki bi vključevale načrt vadbe in nasvete o zdravem načinu življenja.

Bi želeli še kaj dodati?

Kadar ne bi uporabljal Apple Watch simulatorja in bi si želel obiskati golf igrišče, bi si želel merjenje razdalj do ovir, razdalj do luknje in razdalj do naslednjega zavoja. Tako bi lahko imel možnost lažje odločitve, katero palico uporabiti. Ampak vaša aplikacija me je kar navdušila in bi jo z veseljem testiral.