KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Vaidotas Bučinskas

Akselerometro tyrimas ir taikymas mobiliųjų telefonų žaidimų programavime

Magistro darbas

Darbo vadovas

dr. Sigitas Drąsutis

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Vaidotas Bučinskas

Akselerometro tyrimas ir taikymas mobiliųjų telefonų žaidimų programavime

Magistro darbas

Recenzentas
doc. dr. Tomas Skersys

2011-05-30

Darbo vadovas
dr. Sigitas Drąsutis

2011-05-30

Atliko
IFM9/4 gr. Stud.
Vaidotas Bučinskas

2011-05-30

Kaunas, 2011

Summary

Research of accelerometer possibilities
for mobile phone games control development

There are more then 5 billion cell phone subscriptions globally today, which is more then 5 times bigger then personal computer users. Most of the cell phones nowadays are more powerful then personal computers back to the 90s. This allows us to use cell phones not only for communication, but also as fully functional multimedia or gaming devices.

Accelerometer in mobile device was introduced not so long ago with the main purpose – automatically control screen rotation based on viewing angle. Nintendo introduced a new great idea to use accelerometer data and gestures for additional user input as control. This idea can be implemented in mobile devices too, but the main problem is how to use raw, noisy, jittery accelerometer data to control something on the screen. This is the main topic of this research.

The main objectives of this research is:

- Accelerometer feasibility study,
- Reasearch the potential of using accelerometer as controller and possible similar systems,
- Create constructive and explicit method to use accelerometer as controller, and use this method to create a simple game prototype.
- Perform experiment with created game prototype to research accelerometer advantages over classic controls.

To achieve these objectives a requirements specification and detailed project for a software system is performed.

The final method features are the following:

- ability to control game object in any direction, use simple gestures,
- smooth game object movement using data filters and rotation angles interpolation,
- ability to calibrate accelerometer data,
- speed and magnitude data.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

1 Įvadas

Pasaulyje yra daugiau nei 5 milijardai mobiliųjų telefonų abonentų, o tai yra 5 kartus daugiau nei personalinių kompiuterių vartotojų. Didžiosios dalies dabartinių mobiliųjų telefonų pajėgumas yra didesnis nei prieš dešimtmetį išleistų personalinių kompiuterių. [1] Tai leidžia mobiliuosius telefonus naudoti ne tik susisiekimui belaidžiu ryšiu, bet ir kaip pilnaverčius multimedijos ar mobilių žaidimų konsolių įrenginius.

2006 metais kompanija Nintendo pristatė revoliucinę tapusią žaidimų konsolę "Wii", kurios labiausiai inotyvi ypatybė – valdymo įrenginys "Wiimote". Šis valdymo įrenginys gali nustatyti ir posūkio kampą, ir poslinkį trimatėje erdvėje. Šios technologinės naujovės buvo pasiektos įmontuoto akselerometro ir šviesos sensoriaus pagalba. Dėl nuolat besiplečiančių technologijų mobiliuose įrenginiuose, t.y. akselerometro, kameros, wi-fi ir kt. juose atsirado galimybė naudoti įnotyvias informacijos įvedimo formas, panašias į Nintendo sukurtą "Wiimote". [2]

Paprastai akselerometras mobiliajame telefone naudojamas tik vaizdui pasukti. Kai pasukame įrenginį iš horizontalios padėties į vertikalią, akselerometras aptinka judesį ir atitinkamai pakeičia atvaizduojamos informacijos poziciją. Taigi, akimirksniu matome tinklalapį visu pločiu, nuotraukas tinkamu formatu, tam tikrą sąrašą platesniu vaizdu arba valdome žaidimą įvairių judesių pagalba.

Kadangi mobilieji įrenginiai įgauna vis didesnį pagreitį mobiliųjų žaidimų srityje, svarbu kuo anksčiau pasiūlyti ką nors originalaus ir įdomaus šioje rinkoje. Privaloma gerai išnagrinėti akselerometro veikimą ir galimybes, o tuo pačiu suprasti ko reikia šiuolaikiniam inovatoriškam vartotojui, kuris orientuojasi į paprastumą.

Darbe bus nagrinėjamas akselerometro veikimas ir ieškoma kuo paprastesnio ir aiškesnio metodo, akselerometro perduodamus duomenis panaudoti žaidimo valdymo programavimui.

Tyrimo tikslas – ištirti akselerometro galimybes, poveikį, įtaką mobiliajame įrenginyje, o tyrimo rezultatus panaudoti naujam žaidimo valdymo akselerometru metodui sukurti, kuris parodytų netradicines įrenginio panaudojimo ir pritaikymo galimybes.

Uždaviniai:

1. Atlikti akselerometro galimybių analizę.
2. Atlikti akselerometro pritaikymo valdymui analizę ir ištirti esančius sprendimus.
3. Sukurti aiškų ir paprastą metodą žaidimo valdymui programuoti, realizuoti žaidimo prototipą panaudojant sukurtą metodą.
4. Atlikti eksperimentą su sukurtu žaidimo prototipu ir patikrinti akselerometro teikiamą naudą.

2 Analizė

2.1 Analizės tikslas

Pagrindinis analizės tikslas – išnagrinėti tyrimo objekto, akselerometro, ypatumus, t.y. jo bendrus veikimo principus, veikimą pačiame mobiliajame įrenginyje, galimą programavimo aplinką, metodus. Svarbu surasti ir palyginti galimus panašius sprendimus, programavimo ypatybes, nes bus projektuojamas ir kuriamas alternatyvus ir daug paprastesnis ir suprantamesnis metodas akselerometro duomenis pritaikyti žaidimo valdymui. Tam taip pat reikalinga ir būsimų vartotojų analizė, padėsianti nuspręsti, kokių konkrečių sąvybių šiam metodui reikia. Analizės metu bus išnagrinėti reikalavimai duomenims bei nefunkciniai sistemos reikalavimai, nustatyti galimi rizikos faktoriai ir pasirinkti galutinio rezultato kokybės kriterijai.

2.2 Tyrimo sritis, objektas ir problema

Tyrimo sritis – akselerometro taikymas mobiliųjų žaidimų valdymo programavime.

Tyrimo objektas – akselerometro įtaisas mobiliuosiuose telefonuose su pritaikytomis priemonėmis jį valdyti bei programuoti.

Tyrimo problemos:

1. Mobiliuosiuose įrenginiuose akselerometras dažniausiai naudojamas tik ekranui pasukti.
2. Sunku ir nepatogu valdyti mobiliuosius žaidimus dėl smulkių prietaiso mygtukų ar nedidelio ekrano.
3. Akselerometro duomenis tiesiogiai naudoti sudėtingo žaidimo valdyme yra neįmanoma dėl jo triukšmo, ribotų valdymo krypčių nustatymo, posūkio kampo trimatėje erdvėje problemų.

2.3 Tyrimo objekto analizė

Paprastai akselerometrą galima būtų vadinti objekto pozicijos erdvėje matavimo priemone. Jis padeda nustatyti kokioje padėtyje esamu momentu yra įrenginys. Tai leidžia akselerometrą panaudoti įvairiose srityse:

- medicinos,
- statybos,
- navigacijos,
- transporto ir kt. [3]

Pastaruoju metu akselerometras pradėtas naudoti įvairiuose nešiojamuose įrenginiuose ir žaidimų valdikliuose. Mobiliuosiuose telefonuose akselerometras iš pradžių buvo skirtas tik ekranui pasukti, kada priklausomai nuo telefono pozicijos, atitinkamai atvaizduojamas turinys išplėstiniu arba suglaustu formatu. Tačiau tobulėjant technologijai ir mažėjant akselerometro dydžiui, jis pradėtas naudoti ir kaip alternatyva informacijos įvedimui į įrenginius. [4]

2.3.1 Akselerometro veikimo principas

Dėl patobulėjusių mikro technologijų, akselerometro įtaisas yra labai nedidelis, todėl jį paprasta diegti net ir mažiausiuose įrenginiuose. [5] Nors mobilųjį įrenginį puikiai galima valdyti ir be akselerometro, tačiau jo pagalba valdymas pasiekia visai kitą lygį. Nebereikalingi fiziniai mygtukai, o vartotojas visą dėmesį gali sutelkti į valdomą objektą ir vartotojo aplinką.

Dėl savo unikalių savybių akselerometras padeda tiksliai nustatyti, kokioje padėtyje yra valdomas prietaisas. Tai atskleidžia neribotas jo panaudojimo galimybes, ypač prietaiso valdyme. [6]

Yra daug skirtingų rūšių akselerometrų, kurie veikia skirtingais metodais, tačiau galutinis rezultatas yra panašus. Tarp labiausiai paplitusių galima paminėti:

- pjezoelektrinį,
- varžos,
- optinį,
- lazerinį,
- magnetinės indukcijos ir kitus. [7]

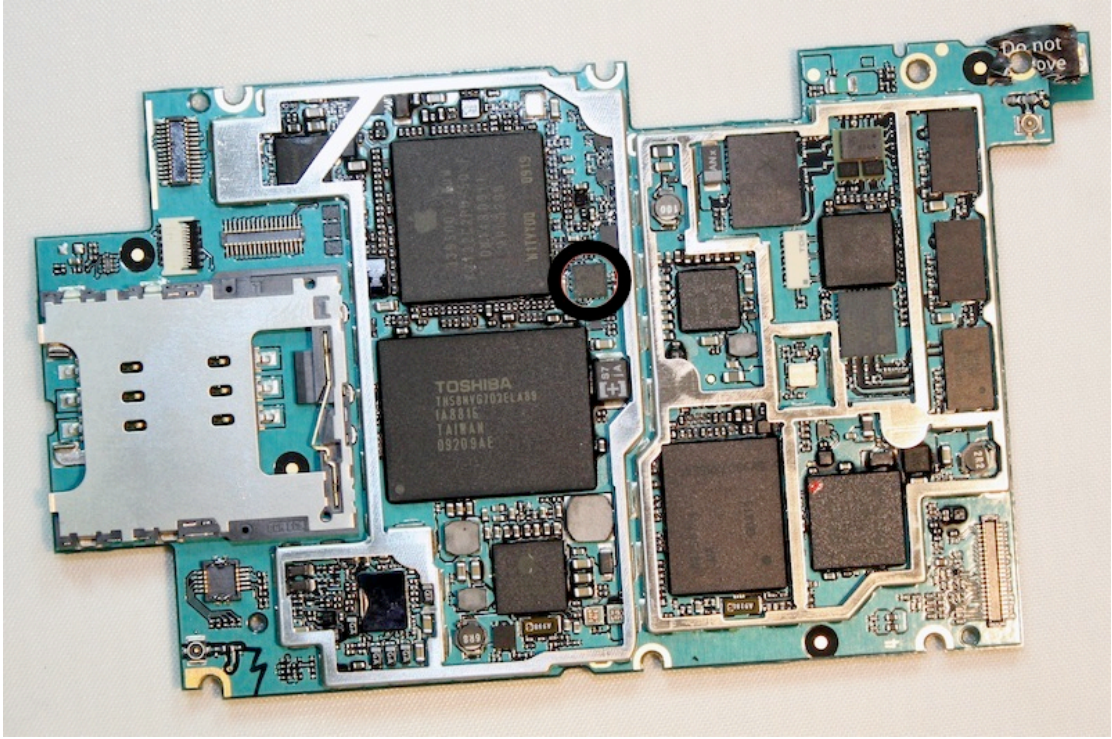
Čia bus aptariamas pjezoelektrinis akselerometras, kadangi dėl savo unikalių savybių bei mažo dydžio jis naudojamas daugelyje mobiliųjų prietaisų, taip pat ir iPhone.

Pjezoelektrinės medžiagos sukuria elektros srovę, kai jos keičia formą, pvz yra suspaudžiamos. Vienaip ar kitaip jos gali būti suspaudžiamos, priklausomai nuo įrenginio padėties. Mobiliuosiuose įrenginiuose naudojami pjezoelektriniai sensoriai dažniausiai yra trijų dimensijų, todėl gali atitinkamus matavimus atlikti kiekvienoje koordinačių plokštumoje. Tokie sensoriai naudoja kvarcinius, silikoninius arba dirbtinius kristalus, kurie gamina elektros energiją spaudžiami, judinami ar lenkiami. [5] Šie kristalai, panašūs į generatorius, dažniausiai yra mikroskopiniai. Priklausomai nuo to, koks kristalų skerspjūvis, jie dažniausiai sukelia elektrinį krūvį kai yra suslėgti konkrečia kryptimi ar palei konkrečią koordinačių plokštumą. Dėl šios priežasties pjezoelektrinis sensorius gali būti labai mažas, o tuo pačiu ir pakankamai tikslus. [8]

Dėl žemo išėjimo signalo ir didelės išėjimo tariamosios varžos, kurią įgyja pjezoelektroninis akselerometras, yra reikalingas sugeneruoto signalo stiprinimas ir tariamosios varžos konversija.

Nors anksčiau tokie stiprintuvai keldavo didelį garsą, tačiau dabar naudojami integrinės grandinės (IC) stiprintuvai, kurie iš karto kartu montuojami ant akselerometro paviršiaus. [9]

Toliau (2.1 pav.) pateikta iPhone mobiliojo telefono sisteminė plokštė, o apskritimu apibrauktas akselerometras. Matome, kad palyginus su visa mobiliojo įrenginio sistemine plokšte, prietaisas yra labai mažas.



2.1 pav. iPhone 3GS akselerometras (ST Microelectronics LIS331DL mikroschema) [10]

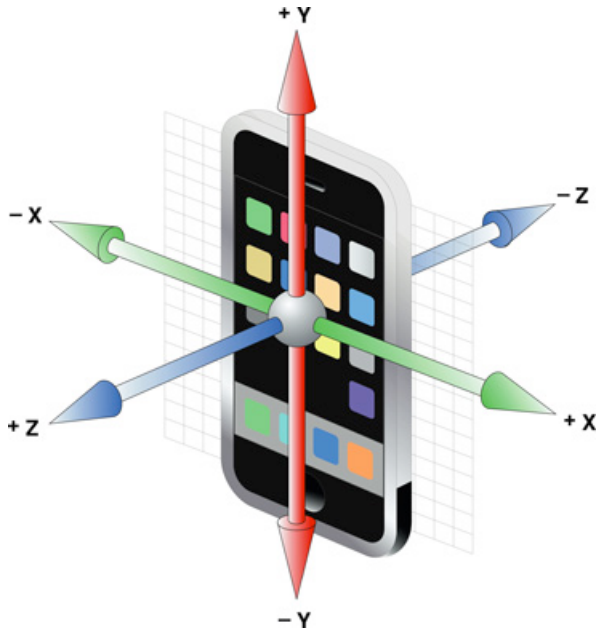
2.3.2 Akselerometras mobiliajame įrenginyje

Kiekvienas akseleracijos įvykis remiasi esamos akseleracijos atvaizdavimu trijuose koordinačių ašyse (kaip parodyta 2.2 pav.). Akseleracijos reikšmės kiekvienai ašiai yra tiesiai perduodamos iš techninės įrangos kaip "G-force" jėgos reikšmės. Taigi, 1.0 reikšmė reprezentuoja maždaug +1g dydžio apkrovą duotąja ašimi, o tuo tarpu -1.0 reprezentuoja -1g dydžio apkrovą. [7]

Praktiškai patikrinus generuojamas reikšmes su AccSim programa, galima tiksliau nusakyti, ką kiekviena koordinatė reiškia (iPhone koordinačių išdėstymas pavaizduotas 2.2 pav.):

- X – atitinka apsisukimą aplink ašį, einančią nuo telefono apačios į viršų, kai telefonas paguldytas horizontaliai ant stalo ekranu į viršų. -1 reikšmė reprezentuoja horizontaliai pastatyto telefono padėtį ekranu į kairę pusę, o 1 – ekranu į dešinę pusę.
- Y – atitinka apsisukimą aplink ašį, einančią nuo kairės iki dešinės telefono šonų, kai telefonas paguldytas vertikaliai ant stalo ekranu į viršų. -1 reikšmė reprezentuoja vertikaliai pastatyto telefono padėtį ekranu į kairę pusę, o 1 – ekranu į dešinę pusę.

- Z – atitinka telefono, paguldyto ant stalo, padėtį ekranu į viršų (-1) arba ekranu į stalą (1).



2.2 pav. iPhone mobiliojo prietaiso koordinačių ašių išsidėstymas [11]

Pritaikius šias akselerometro savybes galima ženkliai pagerinti įrenginio naudojimą. Pagrindinės numatytosios savybės – fotografijų, video peržiūros, elektroninio pašto, trumpųjų žinučių, naršyklės, muzikos grotuvo vaizdo pasukimas. Toliau 2.1 lentelė. pateikiamas numatytų programų papildomų funkcijų ar jų išplėtimo sąrašas, pasukus įrenginį į horizontalią padėtį.

2.1 lentelė. Papildomos iPhone funkcijos ar jų išplėtimas pasukus įrenginį

Programa	**Papildomos, išplėstos funkcijos**
Messages	Didesnė klaviatūra, leidžianti sparčiau rašyti, platesnis žinučių sąrašas.
Safari	Didesnė klaviatūra, leidžianti sparčiau rašyti, tinklalapis pateikiamas platesnėje erdvėje, todėl gali būti aiškiau atvaizduojamas.
iPod	Vietoje sąrašo, kiekvienas muzikos kūrinys atvaizduojamas trimačio paveikslėlio rėmuose, o tai leidžia vizualiai įsiminti kūrinius arba atlikti greitą paiešką pagal vaizdą.
Mail	Didesnė klaviatūra, leidžianti sparčiau rašyti, paprastesnė elektroninio laiško peržiūra.
Photos	Nuotraukos peržiūros metu, priklausomai nuo nuotraukos pozicijos (horizontali ar vertikali), leidžia paprasčiau ir patogiau ją atvaizduoti.
YouTube	Video vaizdas pasukamas visu pločiu suteikdamas “plataus ekrano”

	vaizdą.

Kaip matome, paprasčiausias vaizdo pasukimas gali paspartinti vartotojo atliekamus darbus konkrečioje programoje, suteikti papildomų funkcijų ir apskritai palengvinti valdymą.

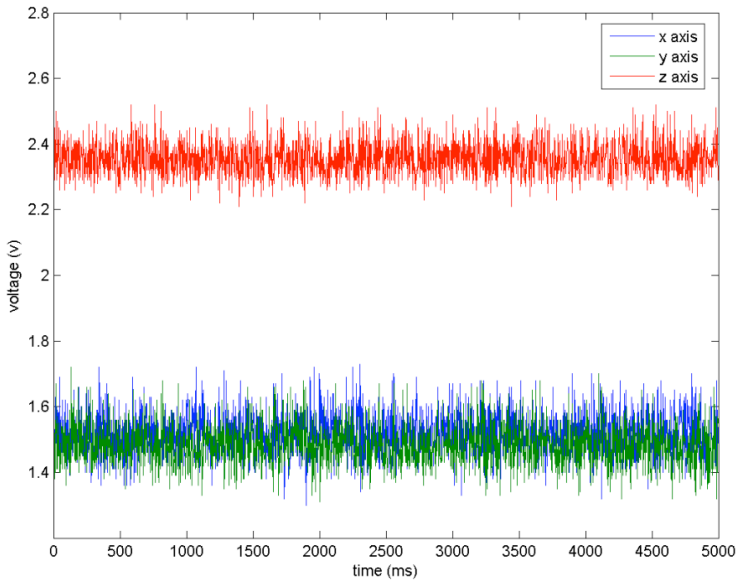
iOS programavimo biblioteka suteikia vaizdo pasukimo galimybę įdiegti ir visuose savo programose.

2.3.3 Akselerometro taikymo problemos

Norint akselerometro duomenis sėkmingai pritaikyti valdymo programavimui, neužtenka vien pasikliauti perduodamais duomenimis iš techninės įrangos. Dirbant su akselerometro duomenimis egzistuoja dvi pagrindinės problemos:

- akselerometro duomenys yra labai triukšmingi ir jautrūs,
- judančio objekto pagal akselerometro duomenis posūkio kampo interpoliacija.

2.3 pav. pateikti akselerometro duomenys x, y ir z ašyse, kai prietaisas padėtas ant stalo ir visiškai nejudinamas. [12] Kaip matome, net ir ramybės būsenoje, akselerometro duomenys yra labai triukšmingi. Tokius duomenis tiesiogiai naudoti žaidimo valdyme būtų neįmanoma, todėl prieš atliekant judesio iteraciją, duomenis reikia filtruoti.



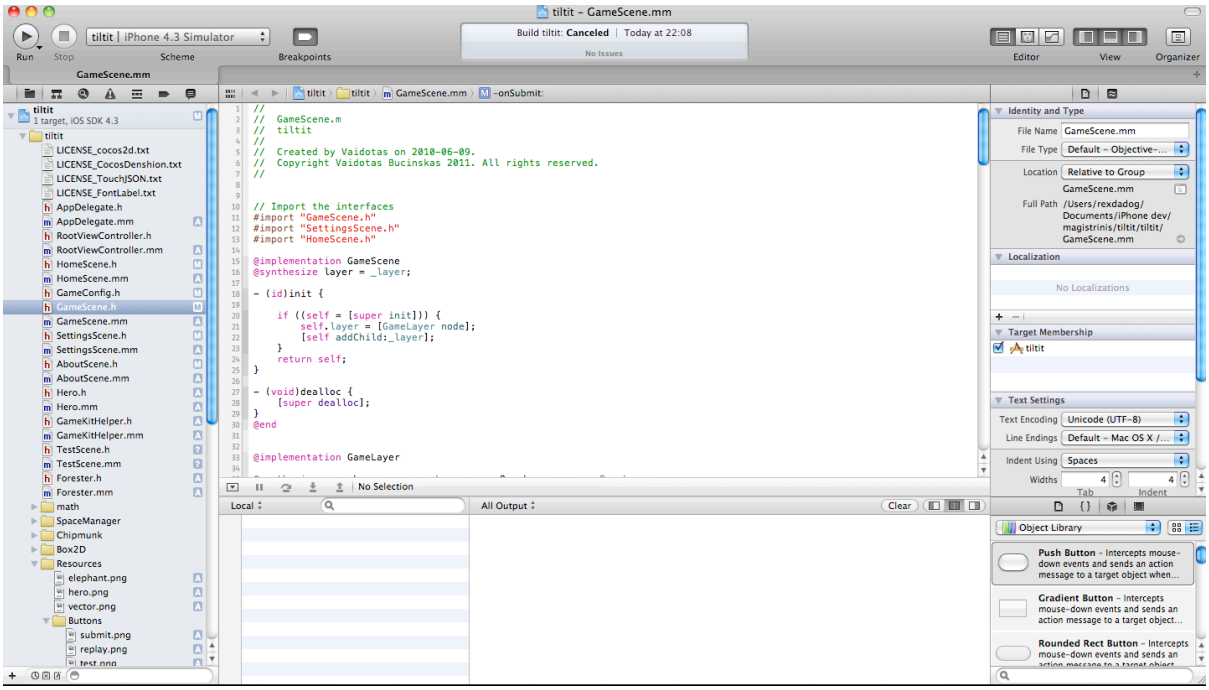
2.3 pav. Triukšmingi akselerometro duomenys [12]

Kita svarbi akselerometro valdomo judančio objekto problema – interpoliacija tarp dviejų judėjimo kampų. Vykstant sukimuisi trimatėje erdvėje visuose x, y ir z ašyse, gali susiformuoti tokia situacija, kada besisukančio objekto sukimosi ašys atsiduria vienoje plokštumoje ir viena iš

besisukančių ašių nebeturi įtakos objekto judėjimui. Ši problema vadinama “Gimbal Lock” problema, kurią galima išspręsti naudojant kvaternionus.

2.3.4 Programavimo aplinka

Visos Mac OS X ir iOS programos kuriamos specializuotoje aplinkoje Xcode (2.4 pav.). Xcode – tai įrankių rinkinys, skirtas kurti programinei įrangai Mac OS X operacinėje sistemoje. Xcode yra nemokamas ir prieinamas visiems Mac OS X vartotojams.



2.4 pav. Xcode 4 projekto langas

Visos programos Mac OS X ir iOS aplinkose rašomos Objective-C programavimo kalba. Tačiau yra galimybė kartu su Objective-C naudoti ir C bei C++ programavimo kalbas nes jos glaudžiai susijusios.

Objective-C yra objektiškai orientuota programavimo kalba, kuri sujungia Smalltalk programavimo kalbos rašymo stilių ir C programavimo kalbą. Objective-C kuria ir aktyviai palaiko Apple. Objective-C yra paremtas C kalbos pagrindu, t.y. C kalba sudaro Objective-C poaibį, todėl Objective-C kompiliatorius supranta ir C kalbą. [14]

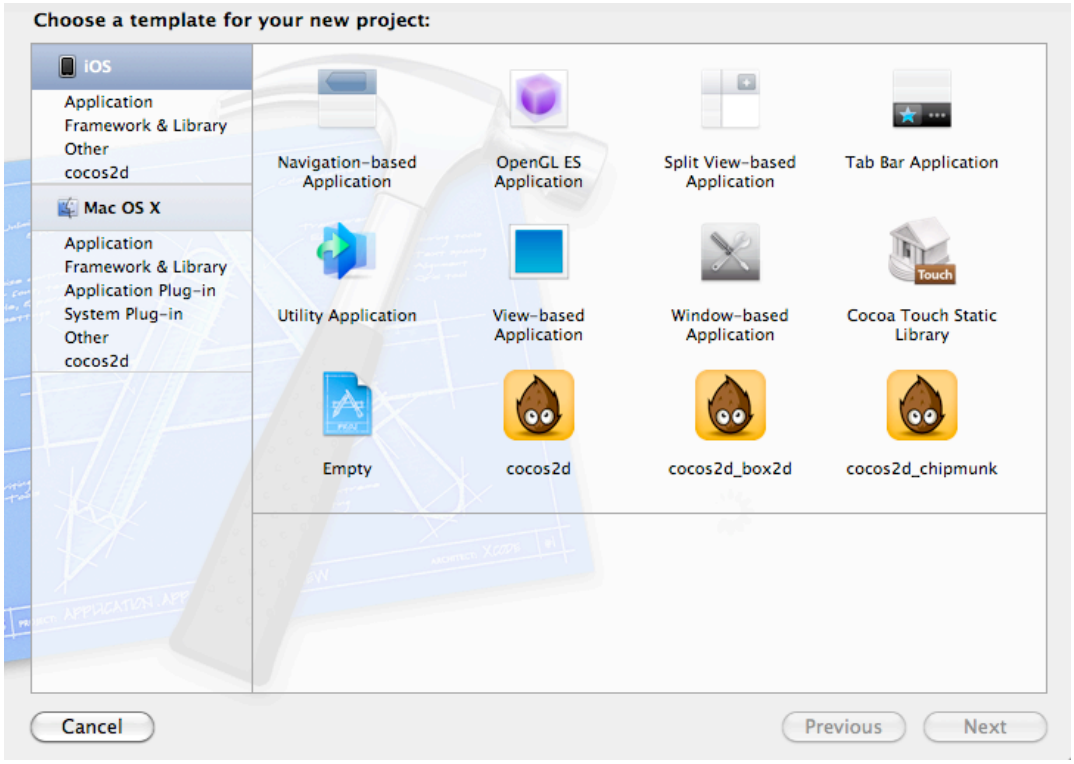
Pagal nutylėjimą, Xcode nepateikia priemonių iOS programavimui, tačiau jas nemokamai galima parsisiųsti iš Apple tinklalapio.

iPhone programos kuriamos naudojant Model-View-Controler (MVC) modelį. MVC modelis visą programos funkcionalumą padalina į 3 dalis:

- Modelį (Model) – klasės, kuriuose laikomi ir paimami programos duomenys,

- Vaizdą (View) – susideda iš langų, valdymo elementų ir kitų elementų, kuriuos mato vartotojas ir su kuriais sąveikauja.
- Valdiklį (Controller) – sujungia modelį ir vaizdą ir aprašo visą programos logiką, kaip elgtis su vartotojo įvedamais duomenimis.

Pagrindinis tokio modelio siekis – objektus, kurie įgyvendina šiuos tris tipus, kiek įmanoma labiau atskirti vienas nuo kito. Tai leidžia paprasčiau, patogiau ir greičiau aprašinėti tam tikras programos dalis, o taip pat panaudoti jas daugiau nei vieną kartą. [13]



2.5 pav. Naujojo projekto pagalbininko langas

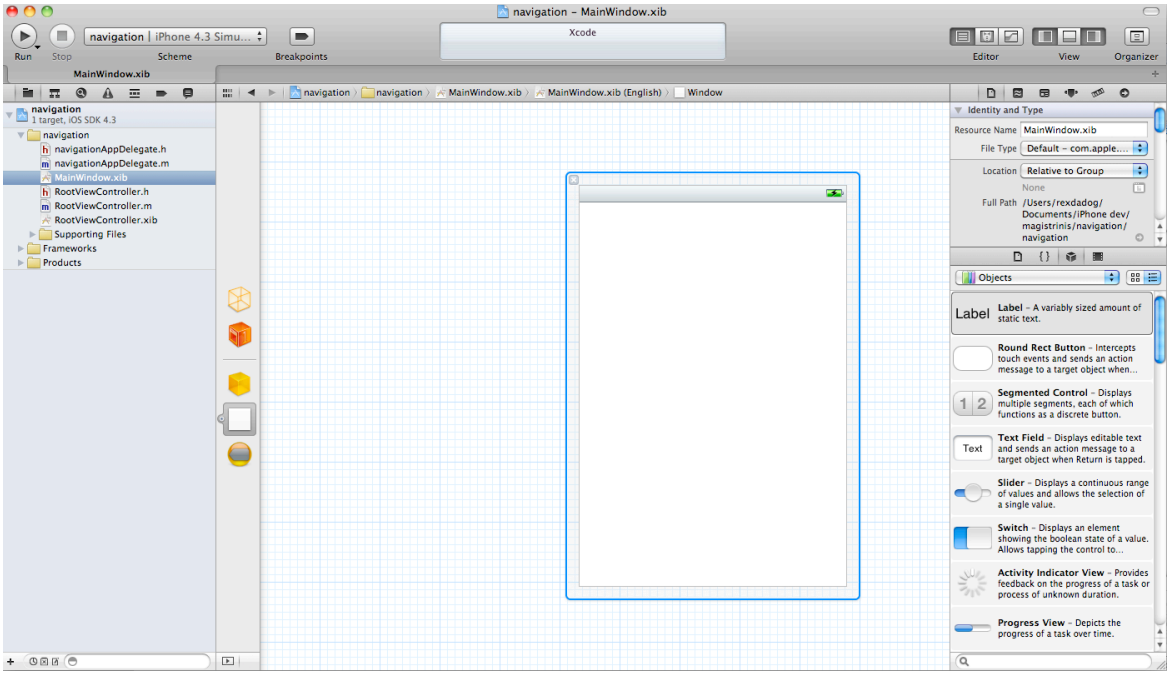
Galimi pradiniai programos programavimo šablonai pavaizduoti 2.5 pav. Kiekvienas šablonas jau iš anksto turi priskirtas bibliotekas ir pradinį programinį kodą programavimo palengvinimui. Pagrindiniai šablonai aprašyti 2.2 lentelė.

2.2 lentelė. Pagrindiniai iPhone programų kūrimo šablonai

Šablonas	**Paskirtis**
Navigation-based	Projektas, paremtas navigacijos valdikliu (controller). Dažniausiai skirtas įvairiems sąrašams ir su jais susijusioms programoms kurti.
OpenGL ES	Projektas, paremtas OpenGL vaizdu (view). Jame pateiktas

	vaizdas, kuriame atvaizduojame OpenGL ES sceną ir laikmatis, skirtas animacijai.
Tab Bar	Projektas, kuris naudos korteles (tabs) . Naudoja kortelių valdiklį ir vaizdo valdiklį su viena pradine kortele.
View-based	Paprastas projektas su tuščiu vaizdu.
cocos2d	Projektas, paremtas cocos2d grafikos varikliuku.

Programos vartotojo sąsaja kuriama naudojant Xcode Interface Builder įrankį (2.6 pav.). Kiekvienas mygtukas, elementas, matomas programos lange, yra tam tikros klasės objektas. Klasių aprašinėti atskirai nereikia, tačiau visada galima jas praplėsti paveldint.



2.6 pav. Interface Builder įrankiai vartotojo sąsajos kūrimui

2.3.5 Programavimo biblioteka

Iš didelės iOS SDK programavimo bibliotekos yra išskirta ir biblioteka, skirta specialiai darbui su akselerometru.

Pagrindinė yra UIAccelerometer klasė, kuri leidžia priimti su akseleracija susijusius duomenis iš techninės įrangos. Kai prietaisas judinamas, jo techninė įranga esančiai programinei įrangai praneša apie tiesinės akseleracijos pokyčius trijų dimensijų erdvėje. Šiuo duomenis galima naudoti ir esamos telefono pozicijos nustatymui (atskaitos taškas – žemė), ir bet kokių momentinių orientacijos pokyčių fiksavimui.

Akselerometro objektas nėra sukuriamas tiesiogiai, o naudojamas jau sukurtas ir bendras visoje sistemoje.

Valdiklyje paskelbiam akselerometro objektą:

```
UIAccelerometer *accelerometer;
```

Jis pasiekiamas per UIAccelerometerDelegate protokolą, kuris ir atsakingas už duomenų srauto valdymą. Akselerometro objektas pradeda siųsti akseleracijos įvykius delegatui iš karto po jo priskyrimo kas numatytą laiko intervalą. Duomenys siunčiami net tada, kai nėra jokio akseleracijos pokyčio, t.y. nuolat, todėl akivaizdžiai labiau naudojama prietaiso energija. Delegato objektas turi iš anksto apibrėžtus metodus, kuriems per numatytus parametrus perduodami duomenys.

Akselerometro inicijavimo pavyzdys [15], čia objektas užregistruojamas gauti akseleracijos reikšmes 60 Hz dažniu:

```
- (void) enableAccelerometerEvents {
        UIAccelerometer *acc = [UIAccelerometer sharedAccelerometer];
        acc.updateInterval = 1/60;
        acc.delegate = self;
}
```

Nurodytu dažniu kviečiamas metodas *accelerometer:didAccelerate* ir akselerometro duomenys trijuose koordinačių ašyse paimami tokiu būdu:

```
- (void) accelerometer:(UIAccelerometer*)accelerometer
              didAccelerate:(UIAcceleration*)acceleration {
        float x, y, z;
        x = acceleration .x; y = acceleration .y; z = acceleration .z;
        // toliau duomenų apdorojimas
}
```

Pradinės akseleracijos reikšmės saugomos UIAccelerometer objekte. Jos visiškai atitinka tai, ką perduoda techninė įranga įrenginiui, t.y. duomenys nėra filtruojami ar kaip nors kitaip keičiami. [13]

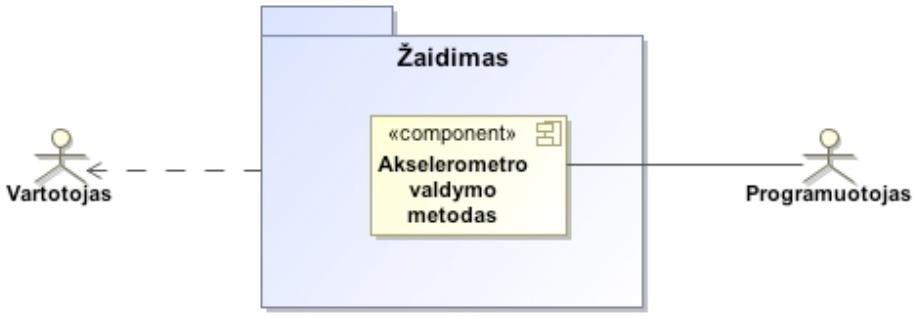
2.4 Vartotojų analizė

2.4.1 Vartotojų aibė, tipai ir savybės

Vartotojų aibę butų galima padalinti pagal tai, kas naudosis sukurtu produktu ir kas bandys pritaikyti darbe kuriamą ir aprašomą metodą akselerometro programavimui. Taigi vartotojų aibę galima padalinti į tokias dvi grupes:

- Vartotojai, besinaudojantys brangesnės grupės mobiliųjų telefonų prietaisais, palaikančiais akselerometro galimybes.
- Programuotojai, suinteresuoti akselerometrą panaudoti savo projektuose, pasitelkdami darbe kuriamą metodą.

Vartotojų principinė schema parodyta 2.7 pav.



2.7 pav. Vartotojų principinė schema

Pagrindinis reikalavimas vartotojams – mobiliojo įrenginio visapusiškas naudojimas, naujovių siekimas ir diegimas. Darbe aprašomas metodas naudojamas žaidimų valdymui, todėl orientuojamasi į žaidimus žaidžiantį vartotoją.

Programuotojas turėtų būti susipažinęs su pagrindinėmis akselerometro galimybėmis, tačiau darbe kuriamas metodas turėtų būti aiškus ir suprantamas nepatyrusiam ar pradedančiajam programuotojui.

2.4.2 Vartotojų tikslai ir problemos

2.3 lentelė. Vartotojų tikslai ir problemos

Problema	**Tikslas**
Vartotojas žaidimą valdo fiziniais mygtukais.	Panaudojus akselerometrą, vartotojo dėmesį sutelkti tik į žaidimo erdvę ir grafinę aplinką, visiškai atitraukiant dėmesį nuo fizinių paspaudimų ir perkeliant valdymą į judesius.
Programuotojui reikia konstruktyvaus metodo, akselerometui programuoti.	Sukurti aiškų ir standartizuotą metodą, akselerometro duomenis panaudoti žaidimų valdymui programuoti.

2.3 lentelė. pateikti vartotojų tikslai ir problemos. Išnagrinėjus akselerometro panaudojimo galimybes būtų galima sukurti naują valdymo metodą ir suprojektuoti užsibrėžtus tikslus atitinkantį žaidimo prototipą, kuris galėtų pakeisti ir palengvinti tai, kaip žaidžiami žaidimai ir valdomas prietaisas.

2.5 Panašių sistemų analizė

Darbo tikslas – sukurti efektyvų, aiškų, nesudėtingą, standartizuotą akselerometru paremtą valdymo metodą. Tam, kad būtų galima patikrinti metodo praktinį pritaikymą ir naudą, jis bus panaudotas žaidimo prototipo kūrime. Taigi, panašių sistemų analizė bus atlikta iš dviejų perspektyvų:

- akselerometro pritaikymo valdymui esamų metodų palyginimas,
- žaidimų, valdomų akselerometru palyginimas.

Akselerometro pritaikymo valdymui esamų metodų palyginimas

Palyginimui buvo ieškomi tokie metodai, kurie labiausiai atitiktų kuriamo metodo specifiką, t.y. būtų orientuoti į valdymą. Metodų palyginimui reikėtų įvertinti šias savybes:

1. pritaikymas valdymui,
2. metodo paprastumas,
3. programavimo galimybės.

Palyginimui buvo pasirinkti tokie pavyzdžiai:

- tiesioginis duomenų naudojimas (AccelerometerGraph) [15],
- gestų identifikavimas (Gesture Recognition) [16],
- fizinio aktyvumo būsenos nustatymas (Activity Recognition) [17][18].

Toliau pateikiamos palyginimo lentelės, padedančios įvertinti kiekvieną iš metodų.

2.4 lentelė. Pritaikymas valdymui

Savybės	AccelerometerGraph	Gesture Recognition	Activity Recognition
Galimybė pritaikyti žaidimo valdymui	Taip	Taip, specifiniam žaidimui	Taip, specifiniam žaidimui
Valdymo kryptys	Bet kokia	Iš anksto apibrėžti gestai	Iš anksto apibrėžtos fizinio aktyvumo būsenos pagal x ašies pokytį
Valdymo sudėtingumas	Paprastas, krypties atpažinimas	Sudėtingas, gestų atpažinimas	Sudėtingas, fizinio aktyvumo atpažinimas

2.5 lentelė. Metodo paprastumas

Savybės	AccelerometerGraph	Gesture Recognition	Activity Recognition
Taikomi algoritmai	Žemų arba aukštų dažnių filtras	Vektoriu kvantavimas, Bajeso teoremos taikymas klasifikavimui, paslėpti Markovo modeliai	Sprendimų lentelės, sprendimų medis, stekas, binarinis medis
Filtravimas	Žemų arba aukštų dažnių filtras	Aukštų dažnių filtras	Filtravimas nenaudojamas

2.6 lentelė. Programavimas

Savybės	AccelerometerGraph	Gesture Recognition	Activity Recognition
Programavimo kalba ir biblioteka	iOS SDK, Objective-C	iOS SDK, C ir Objective-C	C
Pateiktas algoritmas matematine išraiška	Ne	Taip	Taip
Pateiktas algoritmo pseudokodas/kodas	Taip	Taip, tik pseudokodas	Ne
Programavimo aplinka	Mac OS X	Mac OS X	Bet kokia
Programavimo sudėtingumas	Nesudėtingas	Sudėtingas	Sudėtingas

Palyginę metodus nuo paprasčiausio (AccelerometerGraph) iki sudėtingiausio (Gesture Recognition), galime daryti išvadą, kad metodo pasirinkimas labiausiai priklauso nuo to, kokios galutinės sistemos yra siekiama. Kiekvienas iš šių metodų skirtingoje situacijoje yra savaip naudingas. Pvz. norėdami sukurti žaidimą, kuriame reikėtų atlikti tam tikrą gestą, pasirinktumėte gestų atpažinimo metodą. Tą būtų galima atlikti naudojant ir tiesioginio duomenų apdorojimo metodą, tačiau galutinis rezultatas nebūtų toks efektyvus. Gestų atpažinimo ir fizinės būsenos nustatymo metodai naudoja sudėtingus matematinius algoritmus, kurių programinis įgyvendinimas yra techniškai sunkus, todėl šie metodai netinka siekiant greito rezultato.

Šiame darbe bus siekiama sukurti tokį metodą, kuris būtų tarpinis variantas tarp paprasto tiesioginių duomenų naudojimo ir sudėtingo gestų atpažinimo metodo.

Žaidimų, valdomų akselerometru palyginimas

Prieš pradedant esamų žaidimų, valdomų akselerometru, analizę, reikia nusistatyti, kokiais kriterijais vadovaujantis jie bus lyginami. Kadangi akselerometro galimybės bus taikomos mobiliojo žaidimo prototipo kūrimui, tai tikintis geriausio rezultato reikėtų palyginti šias egzistuojančių žaidimų savybes:

1. pasirinkto valdymo metodo įvertinimas, patogumas,
2. akselerometro pritaikymo idėja,
3. aktualumas,
4. galutinis pasitenkinimas.

Žaidimas “Real Racing”

- Valdymas – valdymas geras ir intuityvus. Yra galimybė nusistatyti kelis valdymo režimus, jautrumą, todėl vartotojas valdymą gali prisitaikyti prie savo norų.
- Idėja – akselerometro pagalba visiškai pakeisti automobilio vairo funkciją. Žaidimas valdomas sukinėjant įrenginį tarsi vairą, ko pasekoje sukeliamas tikro vairavimo įspūdis.
- Aktualumas – automobilio žanro žaidimai visada buvo vieni populiariausių. Šiuo atveju akselerometras valdymui tinka idealiai.
- Galutinis pasitenkinimas – akselerometras pritaikytas puikiai. Tai yra viena geriausių žaidimo sričių, kurioje galima sėkmingai pritaikyti akselerometrą.

Žaidimas “Labyrinth 3D”

- Valdymas – valdymas intuityvus, labai tinkantis šiam žaidimui. Rutuliukas rieda pagal fizikos dėsnius, tarsi iš tikrųjų įmestas į dėžutę. Yra galimybė rutuliuką net pašokdinti atitinkamai atliekant staigų judesį į viršų.
- Idėja – akselerometro pagalba valdomas rutuliuko judėjimas, o pagrindinis tikslas – kuo greičiau išridenti rutuliuką iš labirinto bandant išvengti atsitiktinių skylių.
- Aktualumas – tokie žaidimai mobiliuosiuose įrenginiuose visada būdavo populiarūs dėl savo paprastumo, tačiau naudojant akselerometrą žaidimas pakilo į visai naują lygį.

- Galutinis pasitenkinimas – žaidimas įtraukiantis ir įdomus. Atrodo, kad akselerometras visada privalėjo čia būti naudojamas.

Žaidimas “Red Bull Air Race”

- Valdymas – šiame žaidime valdymas yra labai nepatogus. Akselerometras per daug jautrus, todėl net menkiausias pajudinimas sukelia lėktuvo nestabilumą.
- Idėja – tai yra skraidymo simuliatorius. Akselerometro pagalba valdomas lėktuvas, o pagrindinis tikslas, kuo greičiau pasiekti finišo tiesiąją.
- Aktualumas – kaip ir automobilio valdymo žaidimuose, taip ir skraidymo – akselerometras valdyme yra labai aktualus. Visgi skraidymo žaidimai nėra mėgstamųjų tarpe.
- Galutinis pasitenkinimas – žaidimas nėra toks įtraukiantis, valdymas sudėtingas, o paprastos lenktynės be priešininkų yra neįdomios.

Žaidimas “Moto X Mayhem”

- Valdymas – valdymas patogus, bet reikia tam tikrų įgūdžių. Taip pat galima nusistatyti akselerometro jautrumą. Tik pradėjus žaidimą, intuityviai aišku ką reikia pasiekti ir kaip laikyti lygsvarą.
- Idėja – akselerometro pagalba valdoma žaidėjo lygsvara ant motociklo. Pagrindinis tikslas, kuo greičiau, nenukritus nuo motociklo, pervažiuoti kalnuotą kelią.
- Aktualumas – kažkada buvo labai populiarus asmeninių kompiuterių žaidimas, valdomas klaviatūros rodyklių klavišais. Visai kita patirtis valdyti motociklininko lygsvarą akselerometru.
- Galutinis pasitenkinimas – žaidimas įdomus. Didžiulė gausa skirtingų žemėlapių su vis kitokių įgūdžių reikalaujančioms kliūtimis labai greitai neatsibosta.

2.7 lentelė. Sistemų palyginimas dešimtbalėje sistemoje

Lyginimo kriterijai	**Real Racing**	**Labyrinth 3D**	**Red Bull Air Race**	**Moto X Mayhem**
Idėja	9	9	7	9
Aktualumas	10	9	7	9
Valdymo patogumas	10	9	4	10

Galutinis pasitenkinimas	10	9	4	9
Vidurkis	9,75	9	5,5	9,25

Kiekvienas žaidimas buvo vertinamas dešimtbalėje sistemoje, kur 10 reiškia, kad atitinkamas kriterijus įgyvendintas puikiai, be didesnių priekaištų, 9 – labai gerai, tačiau yra nedidelių niuansų ir t.t.

1. **Valdymas.** Savybė, nusakanti ar patogu valdyti atitinkamą žaidimo objektą. Ji buvo įvertinta remiantis asmenine patirtimi atlikus kiekvieno žaidimo testavimą. Automobilio valdymas Real Racing žaidime buvo beveik be priekaištų, o štai Red Bull Air Race – labai nepatogus.
2. **Idėja.** Savybė, reikalinga įvertinti kuo įdomesnį akselerometro pritaikymą žaidimuose. Beveik visų lyginamų žaidimų idėja buvo gera, tačiau tam, kad galutiniai produktai būtų be priekaištų, trūko tam tikrų savybių. Arčiausiai dešimtuko buvo Real Racing dėl labai intuityvaus ir gero valdymo. Visiems kitiems patogaus valdymo labiausiai ir trūko, tačiau dėl savo neblogos idėjos jie tai kompensuoja.
3. **Aktualumas.** Savybė, pagal kurią vertinama peržvelgus pirkimo statistiką ir tai, ko šiuolaikinis vartotojas labiausiai pageidauja. Šioje kategorijoje automobilių simuliatoriai yra labiausiai trokštama prekė.
4. **Galutinis pasitenkinimas.** Savybė, nusakanti vartotojo galutiniu pasitenkinimu produktu po jo testavimo. Įvertinta taip pat remiantis asmenine patirtimi kartu su kitų vartotojų atsiliepimais ir įvertinimais (kiekviena programa turi savo vertinimo statistiką internetinėje parduotuvėje).

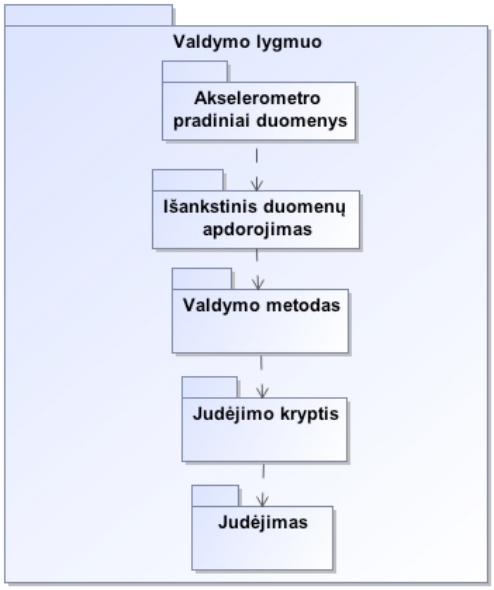
Galutiniai rezultatai, pateikti susumavus visus tarpinius rezultatus ir išvedus jų vidurkį.

Išanalizavus pagrindinius kriterijus galime išskirti esminius reikalavimus sėkmingam žaidimui:

- gera idėja, kur akselerometro reikalingumas būtų akivaizdus,
- patogus valdymas su galimybe vartotojui pačiam nusistatyti akselerometro jautrumą,
- paprastas – sudėtingas žaidimas bus numestas po pirmo bandymo, todėl svarbu išlaikyti jo paprastumą, bet paprastumo kaina negalima sumažinti jo įdomumo.

2.6 Architektūros ir galimų įgyvendinimo priemonių variantų analizė

Principinė galima akselerometro valdymo architektūra pavaizduota 2.8 pav.



2.8 pav. Akselerometro valdymo architektūra

Architektūra susideda iš tokių pagrindinių žingsnių:

- akselerometro pradinių duomenų gavimas –
  akselerometro pradiniai duomenys ateina iš techninės įrangos. Šie duomenys šiuo momentu dar negali būti taikomi objekto valdymui, nes yra neapdoroti.
- gautų duomenų apdorojimas ir pritaikymas tolesniam valdymui –
  šiame žingsnyje taikomas duomenų filtras. Egzistuoja įvairūs filtravimo algoritmai, tačiau šiame darbe bus patikrinti aukštų ir žemų dažnių filtrai, kurie geriausiai tinka tokio tipo duomenims filtruoti. Čia taip pat bus bandoma išspręsti “Gimbal lock” problemą naudojant kvaternionus posūkio kampui skaičiuoti
- valdymo metodo pritaikymas judėjimo krypčiai nustatyti –
  šiame žingsnyje bus sukurtas metodas judinamo žaidimo objekto krypčiai nustatyti, pasitelkus jau tvarkingus ir apdorotus akselerometro duomenis.
- judėjimo krypties panaudojimas objekto judėjimui –
  šiame žingsnyje gauta judėjimo kryptis bus pritaikyta žaidimo objektui animuoti judėjimo kryptimi.

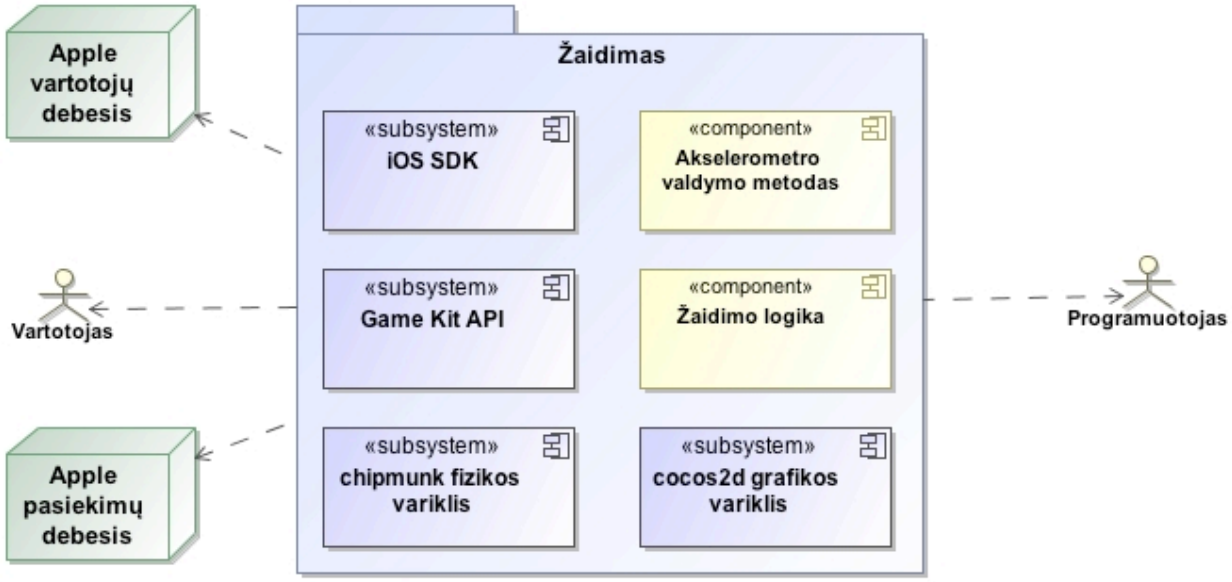
2.7 Siekiamos sistemos apibrėžimas

Siekiamas sprendimas susideda iš kelių atskirų dalių:

- Konstruktyvaus metodo žaidimo valdymui akselerometru sukūrimas, standartizavimas ir aprašymas.
- Metodo pritaikymas žaidimo prototipo programavimui.

Metodo kūrimas – pagrindinis darbo siekis. Metodas turi aiškiai apibrėžti ir standartizuoti akselerometu paremtą valdymą. Jo sėkmingu sukūrimu bus paremtas ir žaidimo prototipo kūrimas, kuris turėtų apimti tokius pagrindinius žaidimų komponentus: rekordo siekimas, bendra rekordų duomenų bazė, rekordų publikavimas, geriausių rezultatų peržiūra. Žaidimas turėtų būti paprastas ir nesudėtingas, parodyti pritaikyto metodo privalumus.

Siekiamos sistemos apibendrintas apibrėžimas pateiktas 2.9 pav.



2.9 pav. Siekiamos sistemos apibrėžimas

Bus siekiama, kad kuriamas akselerometro valdymo metodas prieš aptartas panašias sistemas įgyvendintų šiuos užsibrėžtus privalumus:

- metodas turi būti standartizuotas, nuoseklus,
- pateikta aiški įgyvendinimo struktūra,
- metodas lengvai programuojamas,
- metodą būtų galima pritaikyti ir sudėtingesniam valdymui, ne tik krypties nustatymui,
- užtikrintas akselerometro problemų pašalinimas (triukšmas, drebėjimas ir "Gimbal lock" problema), o tai leistų valdymą padaryti sklandesnį ir malonesnį,

2.8 Reikalavimai duomenims

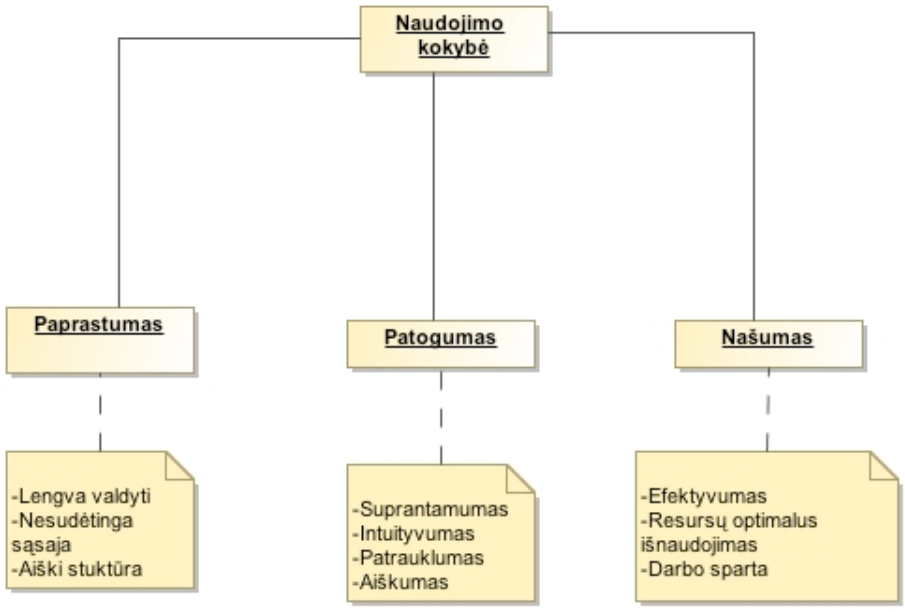
Pagrindinis duomenų šaltinis – akselerometro duomenys. Akselerometro duomenys, perduodami iš akselerometro techninės įrangos į programinę įrangą, yra labai triukšmingi ir nepastovūs, todėl šie duomenys turėtų atitikti tokius reikalavimus:

- Kuo mažiau triukšmingesni akselerometro duomenys.
- Posūkio kampo interpoliacinis apskaičiavimas.

- Žaidimo objekto inercijos apribojimai.

2.9 Nefunkciniai reikalavimai ir apribojimai

Nefunkcinių reikalavimų diagrama pateikta 2.10 pav.



2.10 pav. Nefunkcinių reikalavimų diagrama

- *Paprastumas.*
  - Lengva valdyti, pasitelkiant akselerometrą.
  - Nesudėtinga vartotojo sąsaja.
  - Aiški struktūra ir išdėstymas.
  - Žaidimas turi būti suprantamas be didesnių pagalbos nagrinėjimų.
- *Funkcionalumas*. Sistema turi atlikti visas funkcijas, kurias aprašo.
- *Patogumas*.
  - Visi žaidimo elementai ir valdymas turi būti intuityvūs.
  - Išvaizda turi būti patraukli ir paprasta, neatitraukianti dėmesio nuo žaidimo.
- *Našumas.*
  - Turi būti efektyviai išnaudojamos prietaiso galimybės.
  - Resursai turi būti išnaudojami optimaliai.

- Žaidimo darbo sparta turi netrukdyti vartotojui.

- *Programinė įranga*. Sistemos kūrimui naudojamos Objective-C programavimo kalba ir iOS SDK programavimo biblioteka. Rekordai saugomi MySQL duomenų bazėje ir prietaisui pateikiami XML formatu. Galutiniam žaidimui reikalinga iOS operacinė sistema naudojama kartu su iPhone ir iPod touch įrenginiais.

2.10 Rizikos faktorių analizė

Kuriamas metodas bus ne tik susijęs su jo sėkmingu veikimu, tačiau ir su tuo, kaip sėkmingai jis gali būti pritaikytas kitų sistemų kūrime. Taigi rizikos faktorius galima suskirstyti į dvi atskiras grupes:

- rizikos faktoriai akselorometro valdymo metodui,
- rizikos faktoriai galutiniam produktui, naudojančiam sukurtą metodą.

Rizikos faktoriai akselerometro valdymo metodui:

- Nepakankamas arba neteisingas duomenų filtravimas galėtų metodą padaryti visiškai nenaudojamu.
- Nepakankamai aiškus metodo veikimo aprašymas metodą galėtų padaryti nenaudojamu.
- Metodas teoriškai aprašomas bet kokiai sistemai, tačiau jo realizacija pateikiama tik Objective-C programavimo kalbai. Atsiranda rizika, prarasti potencialius kitų sistemų vartotojus.

Rizikos faktoriai galutiniam produktui, naudojančia sukurtą metodą:

- Nebus naudojamas automatinis akselerometro kalibravimas, o bus leidžiama tai atlikti nustatymų lange pačiam. Čia daroma prielaida, kad nustačius kalibravimą automatiškai, vartotojas nežinos, kaip iš tikrųjų veikia sistema.
- Pateikiamas realizuotas pavyzdys yra labai konkreti metodo panaudojimo iliustracija. Norint metodą pritaikyti savo konkrečiai sistemai, reikalinga papildoma analizė.

2.11 Rezultato kokybės kriterijai

Norint įvertinti sukurto metodo patikimumą, vertę, reikia pasirinkti ir įvertinti atitinkamus kokybės kriterijus. Kokybę vertinsime pagal tokias pasirinktas kategorijas: veikimas, pritaikymas, paprastumas, patikimumas, plečiamumas ir tęstinumas.

Toliau pateikiamos kiekvienos kategorijos kriterijų lentelės ir jų svarba balais nuo 1 (nesvarbus) iki 5 (labai svarbus).

2.8 lentelė. Veikimo kriterijai

Kriterijus	Svarba
Akselerometro duomenų triukšmo panaikinimas	5
Teisingas posūkio kampo nustatymas	5
Teisingas judėjimo krypties nustatymas	5
Sklandus galutinis objekto judėjimas ekrane	5

2.9 lentelė. Pritaikymo kriterijai

Kriterijus	Svarba
Objektą galima judinti visomis kryptimis	5
Objektą galima valdyti imituojant vairo sukimą	4
Objektą galima valdyti gestų pagalba	2

2.10 lentelė. Patikimumo kriterijai

Kriterijus	Svarba
Objektas visada juda ten, kur tikimasi	5
Pritaikius akselerometro duomenų filtravimą, visiškai panaikinamas triukšmas	4
Metodas naudoja efektyvų atminties valdymą	3

2.11 lentelė. Plečiamumo, tęstinumo kriterijai

Kriterijus	Svarba
Teoriškai aprašytas metodas pritaikomas bet kokiai sistemai	5
Sukurtas valdymo metodas paveldimas ir tokiu būdu praplečiamas	4
Sukurtas valdymo metodas neriboja valdymo krypties	3
Yra galimybė keisti jautrumo, filtravimo kalibravimo parametrus	3
Kodas aiškus ir struktūrizuotas	3

2.12 Analizės išvados

1. Ištyrus akselerometro galimybes išsiaiškinta, kad dėl unikalių akselerometro savybių ir savo mažo dydžio jis dažniausiai naudojamas įvairiuose mobiliuosiuose įrenginiuose, tuo pačiu ir mobiliuosiuose telefonuose.
2. Išsiaiškinta, kad egzistuoja dvi svarbios akselerometro panaudojimo problemos: triukšmingi akselerometro duomenys ir pasukimo kampo interpoliacijos sunkumai, atsirandantys dėl "Gimbal lock" problemos.
3. Išnagrinėti panašūs valdymo metodai parodė, kad kiekvienas metodas gali būti savaip sėkmingai pritaikytas skirtingose situacijose.
4. Išbandžius kelis egzistuojančius žaidimus, nustatyta, kad svarbiausios sėkmingo žaidimo savybės yra: idėja, patogus valdymas ir paprastumas.
5. Pasirinkti kokybės kriterijai suskirstyti į kategorijas: veikimas, pritaikymas, paprastumas, patikimumas, plečiamumas ir tęstinumas. Šie kriterijai padės nustatyti, ar sukurtas metodas yra vertingas ir patikimas.
6. Nustatyta, kad rizikingiausias yra ne sukurto metodo naudojimas, o šio metodo taikymas jau galutiniam produktui kurti.
7. Vienintelė iPhone programų kūrimui naudojama programavimo aplinka yra Xcode, o programuojama Objective-C programavimo kalba naudojant iOS SDK biblioteką. Šios bibliotekos pagrindinė klasė, skirta akselerometro valdymui, yra UIAccelerometer.

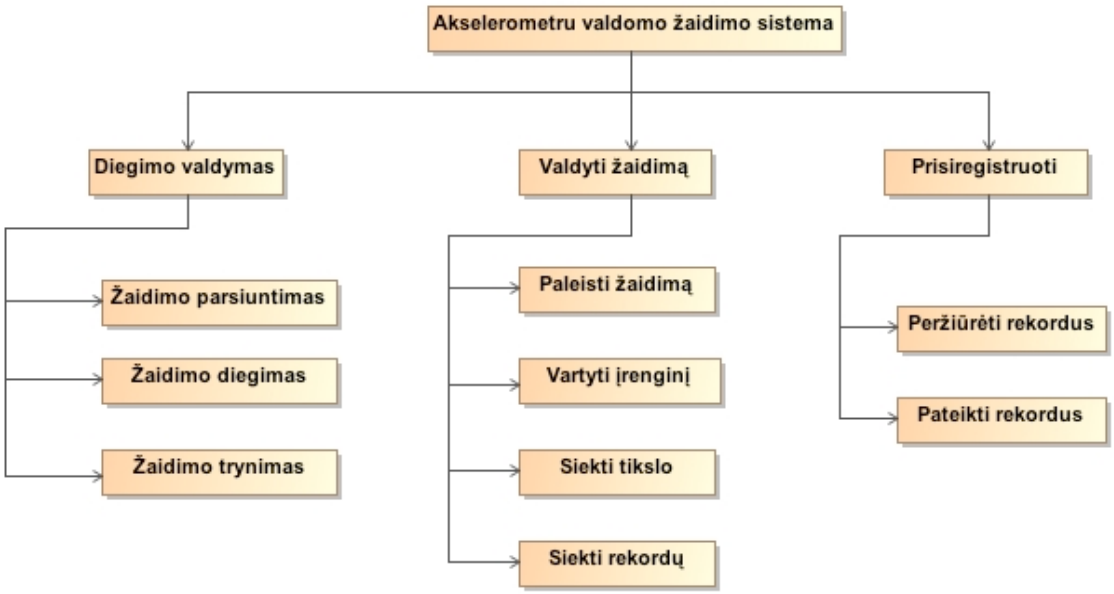
3 Reikalavimų specifikacija ir analizė

3.1 Reikalavimų specifikacija

Akselerometru valdomo žaidimo reikalavimų specifikacija sudaryta remiantis 2 punkte atlikta analize ir jos išvadomis.

3.1.1 Funkciniai reikalavimai

Sistemos funkcinės hierarchijos modelis pateiktas 3.1 pav. Šis modelis hierarchiškai atvaizduoja sistemos funkcijas, kurios taip pat pateikiamos panaudojimo atvejais.



3.1 pav. Akselerometru valdomo žaidimo funkcinės hierarchijos modelis

Funkciniams reikalavimams modeliuoti taikoma panaudojimo atvejų metodika. Kokias funkcijas turi atlikti sistema plačiau apibūdina panaudojimo atvejų aprašymai, o jų veiklos diagramos parodo, kaip šios funkcijos turi būti atliekamos. Akselerometru valdomo žaidimo sistemos panaudojimo atvejai pavaizduoti 3.2 pav.

- Akselerometru valdomo žaidimo panaudojimo atvejai

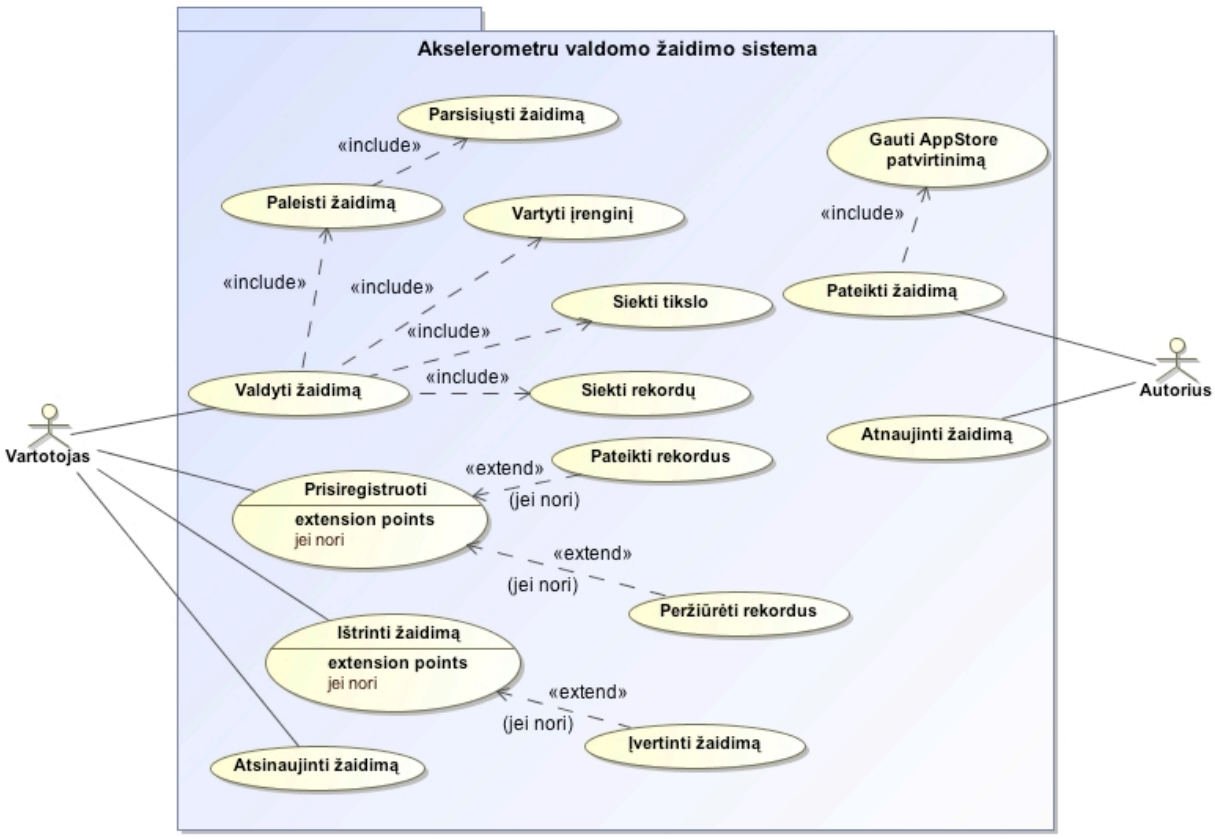
Egzistuoja 2 sistemos dalyviai:

  - Vartotojas – pagrindinis dalyvis, į kurį orientuota visa sistema;
  - Autorius – sistemos kūrėjas.

Pagrindiniai vartotojo panaudos atvejai: valdyti žaidimą, prisiregistruoti, ištrinti žaidimą ir atsinaujinti žaidimą.

Pagrindiniai autoriaus panaudos atvejai: pateikti žaidimą ir atnaujinti žaidimą.

Panaudos atvejai detaliau nagrinėjami 3.1 lentelė. – 3.6 lentelė.



3.2 pav. Akselerometru valdomo žaidimo panaudojimo atvejai

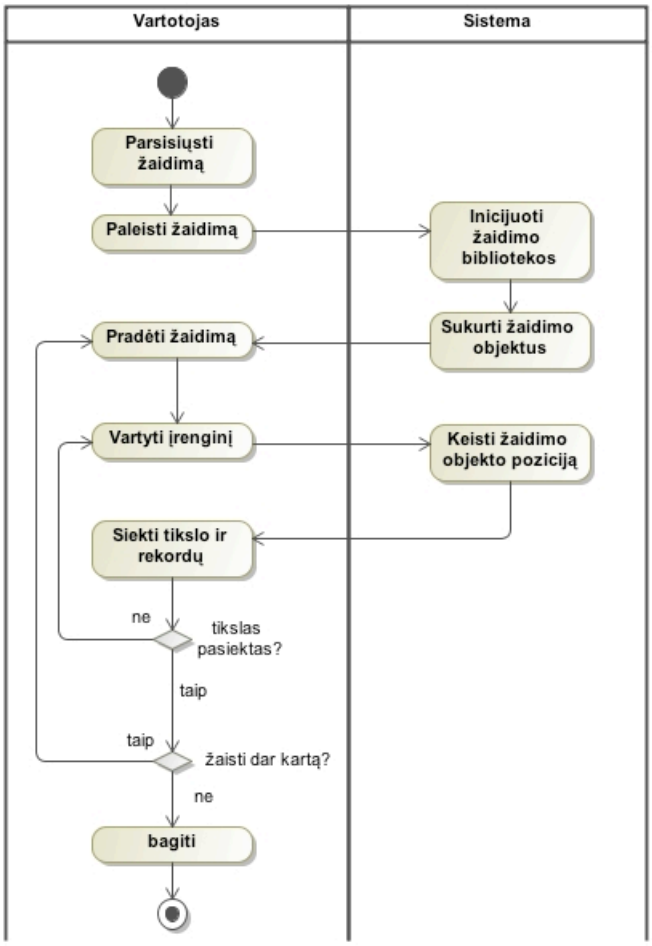
- Panaudojimo atvejų (PA) specifikacijos

PA "Valdyti žaidimą" aprašymas pateiktas 3.1 lentelė., o jo veiklos diagrama pateikta 3.3 pav. Vartotojas jau iš anksto įrenginyje turi žaidimą, jį paleidžia ir sukinėdamas įrenginį žaidžia. Vartotojo siekia tam tikro tikslo ir gauna tam tikrą įvertinimą, kuris, priklausomai nuo pasiekto rezultato, patenka arba nepatenka tarp rekordų.

3.1 lentelė. PA "Valdyti žaidimą"

Panaudojimo atvejis "Valdyti žaidimą"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	Vartotojas turi žaidimą savo įrenginyje
Sužadinimo sąlyga	Žaidimas paleidžiamas
Susiję panaudojimo atvejai	*Apima:* Paleisti žaidimą, Valdyti įrenginį, Siekti tikslo, Siekti rekordų.
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas paleidžia žaidimą	1.1 Žaidimas paleidžiamas, inicijuojamos žaidimo bibliotekos 1.2 Sukuriamas žaidimo objektas
2.1 Vartotojas pradeda žaidimą 2.2 Vartotojas sukinėja įrenginį	2.1 Žaidimo objektas reaguoja į pasukimus ir keičia savo poziciją
3. Vartotojas siekia tikslo	3.1 Pateikiamos užduotys, kurias reikia įvykdyti

4. Vartotojas siekia rekordų	4.1 Pateikiami geriausi rezultatai
5. Vartotojas pasiekia tikslą ir pasirenka ar žaisti, ar baigti darbą	5.1 Žaidimas pradedamas iš naujo arba baigiamas
Po sąlyga	-
Alternatyvūs scenarijai	
-	



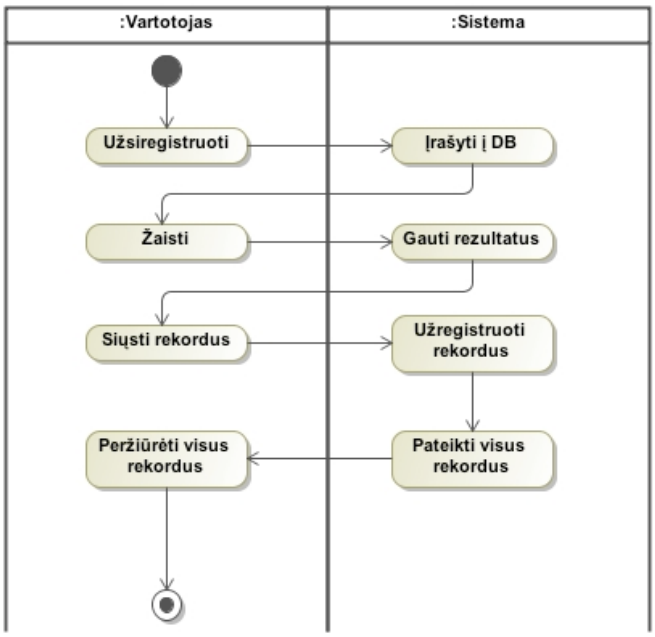
3.3 pav. Panaudojimo atvejo "Valdyti žaidimą" veiklos diagrama

PA "Prisiregistruoti" pateikiamas 3.2 lentelė., o jo veiklos diagrama pateikiama 3.4 pav. Kad galėtų pateikti ir peržiūrėti žaidimo rekordus, vartotojas privalo būti užsiregistravęs sistemoje. Kiekvieną kartą, pasiekęs naują savo rekordą, jis gali siųsti rezultatą į žaidimo serverį ir ten jį užregistruoti. Iš sistemos tada gaunamas geriausių rezultatų sąrašas, kurį galima palyginti su savo rezultatais.

3.2 lentelė. PA "Prisiregistruoti"

Panaudojimo atvejis "Prisiregistruoti"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	Vartotojas neužsiregistravęs

Sužadinimo sąlyga	Paspaudžiamas registracijos mygtukas
Susiję panaudojimo atvejai	*Išplėtimas:* Pateikti rekordus, Peržiūrėti rekordus.
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas užsiregistruoja	1.1 Vartotojas įrašomas į registruotų vartotojų zoną
2. Vartotojas žaidžia žaidimą	2.1 Gaunami tam tikri rezultatai
3. Vartotojas siunčia savo rekordus	3.1 Sistema užregistruoja rekordą
4. Vartotojas peržiūri visus rekordus	4.1 Sistema pateikia visus rekordus
Po sąlyga	Žaidėjas prisiregistravęs
Alternatyvūs scenarijai	
-	



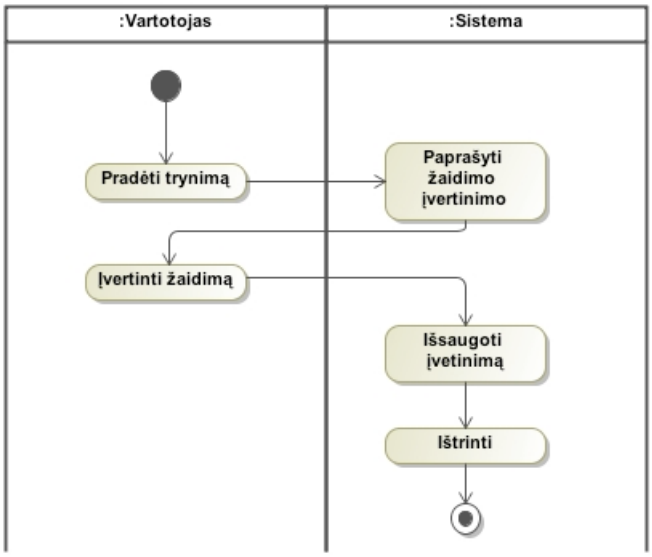
3.4 pav. Panaudojimo atvejo "Prisiregistruoti" veiklos diagrama

PA "Ištrinti žaidimą" pateiktas 3.3 lentelė., o jo veiklos diagrama – 3.5 pav. Vartotojas, pradėdamas trynimo veiksmą, gauna pranešimą apie programos įvertinimą. Jis gali žaidimą įvertinti, arba atsisakyti įvertinimo ir tęsti programos trynimą. Galutinis įvertinimas vėliau matomas programų talpinimo sistemoje. Tai padeda kitiems vartotojams apsispręsti, ar parsisiųsti programą į savo įrenginį.

3.3 lentelė. PA "Ištrinti žaidimą"

Panaudojimo atvejis "Ištrinti žaidimą"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	Vartotojas turi parsisiuntęs žaidimą
Sužadinimo sąlyga	Paspaudžiamas trynimo mygtukas
Susiję panaudojimo atvejai	*Išplėtimas:* Įvertinti žaidimą.

Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas ištrina žaidimą	1.1 Pateikiamas pasiūlymas įvertinti žaidimą
2. Vartotojas įvertina žaidimą	2.1 Sistema išsaugo įvertinimo duomenis 2.2 Žaidimas ištrinamas iš sistemos
Po sąlyga	Žaidimas ištrintas iš sistemos
Alternatyvūs scenarijai	
-	

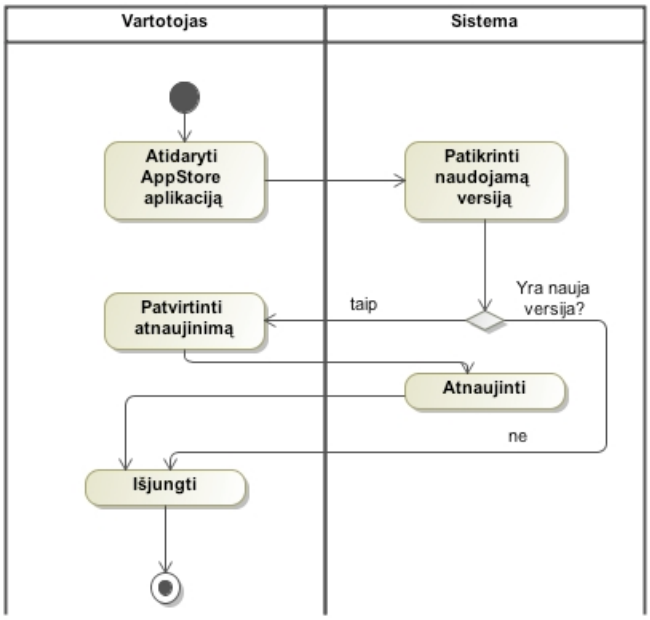


3.5 pav. Panaudojimo atvejo "Ištrinti žaidimą" veiklos diagrama

PA "Atsinaujinti žaidimą" pateiktas 3.4 lentelė., o jo veiklos diagrama – 3.6 pav. Atsinaujinimo procesas pradedamas AppStore programoje, kuri automatiškai patikrina, ar neegzistuoja nauja žaidimo versija ir pasiūlo atsinaujinti. Žaidimas parsiunčiamas į įrenginį ir automatiškai įdiegiamas.

3.4 lentelė. PA "Atsinaujinti žaidimą"

Panaudojimo atvejis "Atsinaujinti žaidimą"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	Vartotojas turi seną žaidimo versiją
Sužadinimo sąlyga	AppStore programoje vartotojas gauna pasiūlymą atsinaujinti žaidimą
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas atidaro AppStore programą	1.1 Sistema patikrina, ar egzistuoja nauja versija ir pasiūlo atsinaujinti
2. Vartotojas patvirtina atnaujinimą	2.1 Sistema atnaujinama
Po sąlyga	Nauja žaidimo versija
Alternatyvūs scenarijai	
-	

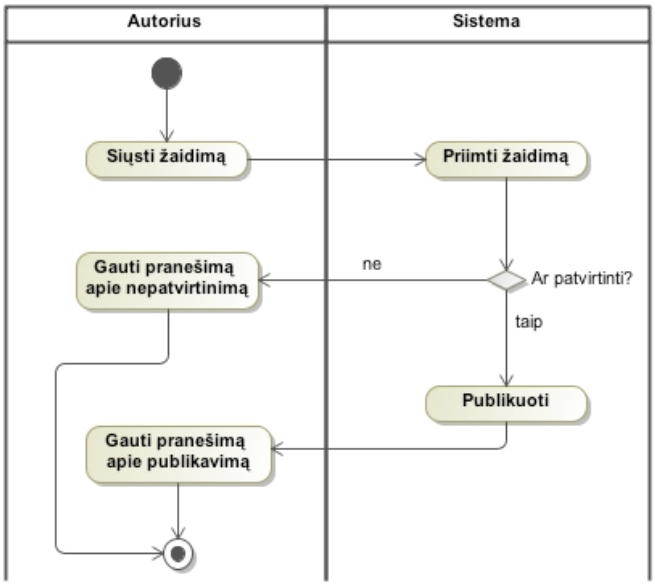


3.6 pav. Panaudojimo atvejo “Atsinaujinti žaidimą” veiklos diagrama

PA “Pateikti žaidimą” aprašytas 3.5 lentelė., o jo veiklos diagrama pateikta 3.7 pav. Žaidimo autorius žaidimą privalo pateikti AppStore sistemai, kuri jį patvirtina arba atmeta. Nuo to priklauso, ar žaidimą bus galima parsisiųsti iš AppStore programos. Tai yra vienintelis legalus būdas įsirašyti bet kokią programą.

3.5 lentelė. PA “Pateikti žaidimą”

Panaudojimo atvejis “Pateikti žaidimą”	
Aktorius	Žaidimo autorius
Prieš sąlyga	Žaidimas neegzistuoja AppStore sistemoje
Sužadinimo sąlyga	Žaidimas siunčiamas į AppStore sistemą
Susiję panaudojimo atvejai	*Apima:* Gauti AppStore patvirtinimą.
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Autorius siunčia žaidimą	1.1 Gaunamas pranešimas apie gautą žaidimą ir jo patvirtinimo būseną
2. Žaidimas patvirtinamas arba atmetamas	2.1 Sistema gali būti parsiunčiama į vartotojų įrenginius 2.2 Žaidimas neužregistruojamas AppStore sistemoje ir jo parsisiųsti negalima
Po sąlyga	Žaidimas užregistruojamas AppStore sistemoje, jį galima parsisiųsti
Alternatyvūs scenarijai	
-	

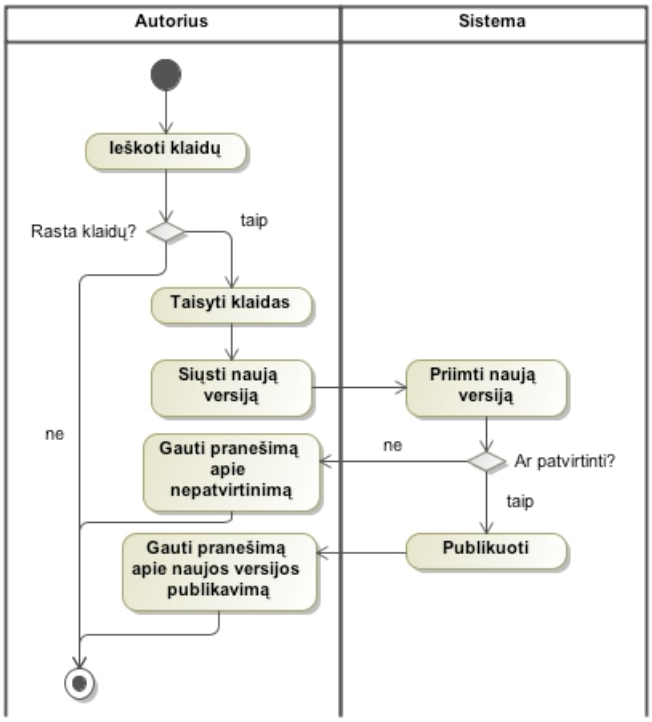


3.7 pav. Panaudojimo atvejo "Pateikti žaidimą" veiklos diagrama

PA "Atnaujinti žaidimą" pateiktas 3.6 lentelė., o jo veiklos diagrama pateikta 3.8 pav. Pastebėjęs klaidą, arba gavęs apie ją pranešimą, žaidimo autorius atlieka tam tikras pataisas. Tokiu atveju žaidimą vėl reikia pateikti į AppStore sistemą tam, kad žaidėjai ją galėtų atsinaujinti. Atnaujinimo pateikimo procesas panašus į pradinį žaidimo pateikimo procesą. Tik gavus patvirtinimą bus galima atsisiųsti žaidimo atnaujinimą.

3.6 lentelė. PA "Atnaujinti žaidimą"

Panaudojimo atvejis "Atnaujinti žaidimą"	
Aktorius	Žaidimo autorius
Prieš sąlyga	Žaidime atsiranda klaidų
Sužadinimo sąlyga	Klaidos taisomos
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1.1 Autorius ieško klaidų 1.2 Autorius randa klaidų žaidime	1.1 Sistema veikia nekorektiškai
2.1 Autorius pataiso klaidas 2.2 Siunčia naują versiją	2.1 Sistema patvirtina naują versiją 2.2 Sistema nepatvirtina naujos versijos
3. Gaunamas patvirtinimas	3.1 Sistema publikuoja naują versiją 3.2 Sistema nepublikuoja naujos versijos
Po sąlyga	Žaidimas su ištaisytomis klaidomis
Alternatyvūs scenarijai	
-	



3.8 pav. Panaudojimo atvejo “Atnaujinti žaidimą” veiklos diagrama

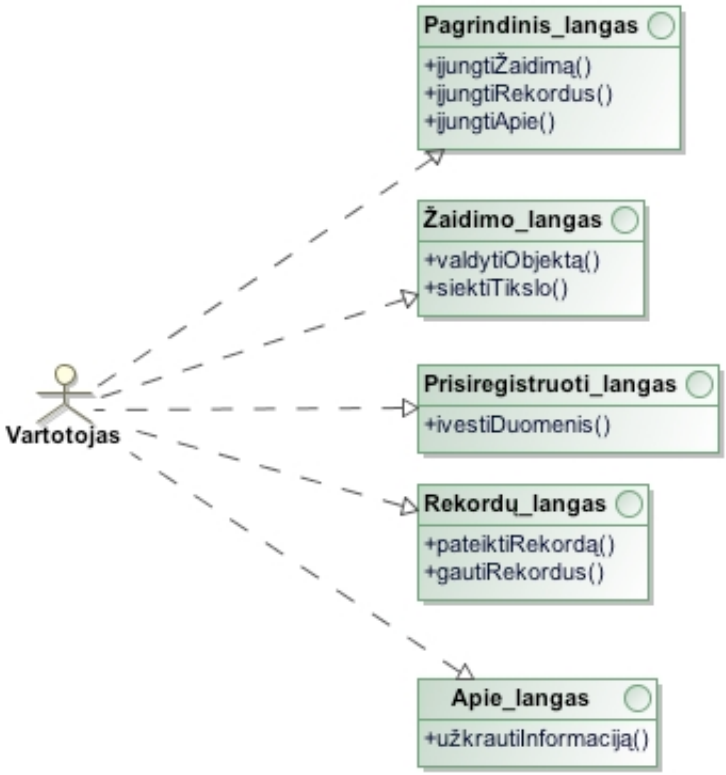
- Vartotojų interfeiso modelis

Vartotojų interfeiso modelis parodo vartotojo bendravimą su žaidime egzistuojančiais informaciją perteikiančiais langais, kurie aprašyti kaip interfeisai. Kiekvienas langas aprašo tame lange egzistuojančias tam tikras funkcijas.

Sistemoje apibrėžti šie pagrindiniai langai:

- Pagrindinis langas,
- Žaidimo langas,
- Prisiregistravimo langas,
- Rekordų langas,
- Apie langas.

Vartotojo sąsaja detaliau aprašyta 4.1 lentelė.

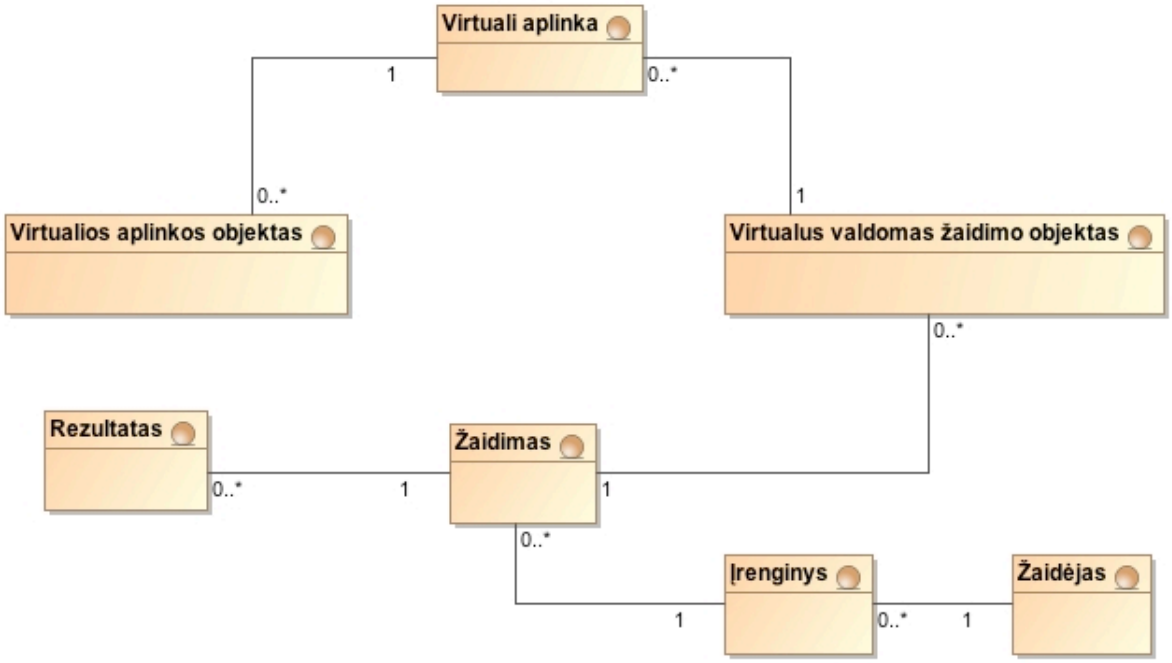


3.9 pav. Sistemos vartotojo interfeiso modelis

3.2 Dalykinės srities modelis

3.10 pav. pateiktas akselerometru valdomo žaidimo sistemos dalykinės srities modelis. Šiame modelyje aprašytos svarbiausios esybės, naudojamos darbe. Modelio kiekvienos esybės paskirtis aprašyta 3.7 lentelė.

Paprastai šį modelį galima suprasti taip: kiekvienas žaidėjas gali valdyti daug įrenginių, kuriuose įrašyta daug žaidimų. Žaidime valdoma daug virtualių žaidimo objektų, kurie patenka į daug virtualių aplinkų, o jos sudarytos iš daug virtualių objektų. Žaidime galima fiksuoti daug rezultatų, kurie sudaro rekordų visumą.



3.10 pav. Dalykinės srities esybių modelis

3.7 lentelė. Dalykinės srities esybės

Esybė	Paskirtis
Žaidėjas	Tai pagrindinis veikėjas, valdantis visą sistemą.
Įrenginys	Priemonė, sistemai valdyti.
Žaidimas	Sistema, kurią valdo žaidėjas.
Virtualus valdomas žaidimo objektas	Apibūdina žaidime valdomą objektą, kuris naudojamas žaidimo tikslams siekti.
Virtuali aplinka	Apibūdina žaidimo aplinką, kurioje vyksta žaidimas.
Virtualios aplinkos objektas	Apibūdina aplinkos objektus, kurie sudaro bendrą aplinką.
Rezultatas	Apibūdina pasiektą žaidimo rezultatą. Rezultatų gausa sudaro rekordų visumą.

3.3 Reikalavimų analizės apibendrinimas

Reikalavimų analizės metu buvo išnagrinėti sistemos panaudojimo atvejai ir detaliai specifikuojami kiekvieno panaudojimo atvejo reikalavimai, apibrėžtas funckinės hierarchijos modelis ir sudarytas dalykinės srities esybių modelis.

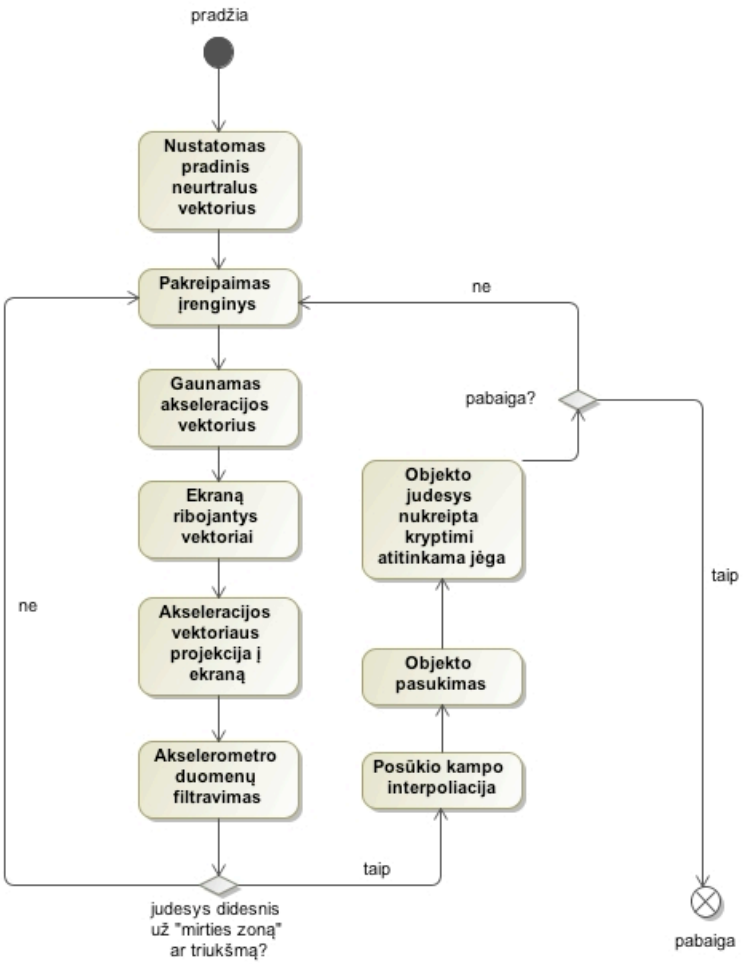
4 Sistemos projektas

Sistemos projekto tikslas yra apibrėžti būsimos sistemos kontūrus, detalizuoti jos veikimą. Projektas detaliai apibrėžia, kaip turi būti realizuoti nustatyti reikalavimai remiantis sistemos logine architektūra, detaliuoju planu, sistemos elgsenos, duomenų bazės ir realizacijos modeliais.

Sistemos projektas taip pat detalizuoja vartotojo sąsają bei apibūdiną jos elementų funkcionalumą. Projekte turi būti iš anksto numatyti realizacijos ir diegimo modeliai, aprašantys sistemos veikimui reikalingus parametrus.

4.1 Metodo pagrindimas ir esmės išdėstymas

Sukurto valdymo metodo principinė algoritmo schema parodyta 4.1 pav.



4.1 pav. Valdymo metodo algoritmas

Metodo veikimo pagrindinė idėja

Labai paprastam ir nesudėtingam žaidimui, nereikalaujančiam tikslumo, akselerometro perduodamus duomenis žaidimo objekto valdymui galima panaudoti tiesiogiai, tačiau siekiant

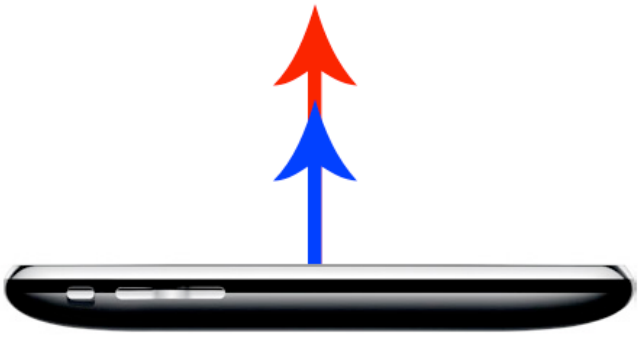
sukurti sudėtingesnį valdymą, produkto kokybės ir universalumo to neužtenka, nes atsiranda tam tikros problemos:

- galima nustatyti tik elementariausias kryptis – į kairę, į dešinę, į viršų ir į apačią,
- tiesioginiai duomenys yra triukšmingi, beveik neįmanoma imituoti tvarkingai judančio objekto,
- negalima nustatyti judėjimo stiprumo, pagreičio, greičio,
- egzistuoja posūkio kampo interpoliavimo problemos,
- skaičiavimas nėra optimalus,
- valdymas tik imituojamas, neatspindi akselerometro, kai valdymo prietaiso.

Kad išvengtume visų šių problemų, buvo sukurtas naujas valdymo metodas, remiantis vektorinės algebros operacijomis trimatėje erdvėje. Šis būdas labiau orientuotas į įrenginį, kaip žaidimo valdymo įrankį.

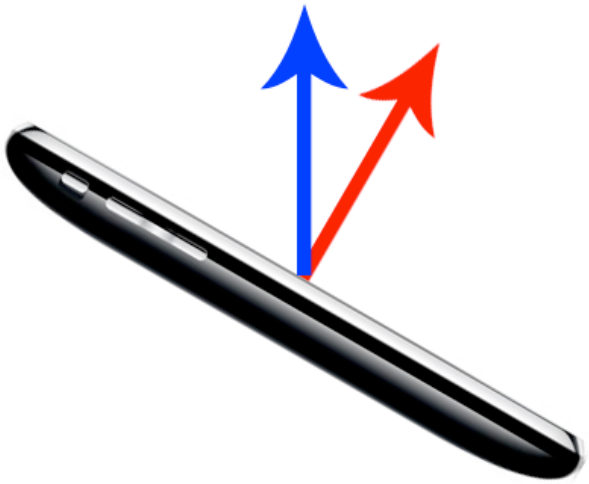
Valdymas paremtas fiziniu įrenginio pakreipimu, o ne jo imitavimu. Tokio veikimo algoritmas grafiškai pavaizduotas 4.2 pav.-4.3 pav. Naudojami 4 pagrindiniai žingsniai judesio krypčiai nustatyti:

1. Turime 2 pradinius vektorius – neutralų (mėlynas) ir invertuotą akselerometro vektorių (raudonas). Pradinis neutralus vektorius visada turi būti nukreiptas į žaidėją. Šis vektorius gali būti fiksuojamas žaidimo metu. Tai atspirties taškas, vektoriaus poslinkiui fiksuoti. Šis vektorius naudojamas prietaiso valdymo kalibravimui. (žr. 4.2 pav.)



4.2 pav. Pradinė įrenginio padėtis

2. Įrenginį pakreipiame tam tikra kryptimi, pvz. 45 laipsniais link savęs. Neutralus vektorius išlieka fiksuotas, o invertuotas akselerometro vektorius juda kartu su įrenginiu. (žr. 4.3 pav.)



4.3 pav. Įrenginys pakreipiamas

3. Šiame žingsnyje apskaičiuojame išilgai ir skersai per ekraną einančius vektorius, į kuriuos formuosime mūsų akselerometro vektoriaus (raudonas) projekciją X ir Y ašyse. Šiems vektoriams apskaičiuoti naudojama vektorinė sandauga. (žr. 4.4 pav.)

Vektorinė sandauga apskaičiuojama pagal formulę:

$$a \times b = ab \sin \theta n$$

4.4 pav. Gauname ekraną ribojančius vektorius

4. Kad galėtume galutinai apskaičiuoti mūsų judėjimo kryptį ir greitį, reikia paskaičiuoti invertuoto akselerometro vektoriaus projekciją į išilgai ir skersai per ekraną einančius vektorius. Vektoriaus projekciją ekrane į anksčiau gautus ekraną padengiančius vektorius apskaičiuojame naudodami skaliarinę akselerometro vektoriaus ir šių vektorių sandaugą. (žr. 4.5 pav.)

Skaliarinė sandauga apskaičiuojama pagal formulę:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

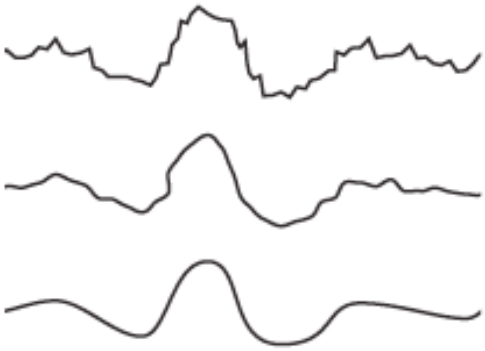
Čia $a = [a_1, a_2, a_3]$ ir $b = [b_1, b_2, b_3]$.

4.5 pav. Suformuojame judesio projekciją į ekraną

Taigi turėdami šį akseleracijos vektorių, galime galiausiai valdyti mūsų žaidimo objektą, nukreipdami jį atitinkama kryptimi ir stiprumu į apačią.

Triukšmingų duomenų filtravimas

Akselerometro pateikiami neapdoroti duomenys yra labai triukšmingi, todėl naudojant tokius duomenis, neįmanoma žaidimo objekto atvaizduoti be drebėjimo. Kad galėtume naudoti akselerometro duomenis objekto judesiui, jo reikšmėms reikia pritaikyti žemų dažnių filtrą. Filtro įtaką akselerometro triukšmingiems duomenims matome 4.6 pav.



4.6 pav. Akselerometro duomenys prieš filtrą ir po žemų dažnių filtro

Patį paprasčiausią žemų dažnių filtrą būtų galima aprašyti taip:

$y_i = \alpha x_i + (1-\alpha) \cdot y_{i-1}$, kur α yra išlyginimo faktoriaus konstanta, o i apibrėžia reikšmę konkrečiu laiko momentu.

Žaidimo objekto pasukimo kampas ir animacija

Žaidimo objektas susideda iš 9 krypčių animacijos, kuri pateikta 4.7 pav. Iš viso žaidime yra 16 skirtingų krypčių, kuriomis gali judėti objektas. Papildomas 7 kryptis galima gauti horizontaliai apsukus jau egzistuojančias kryptis.

4.7 pav. Žaidimo objekto animacija

Pagrindinė problema, nustatyti žaidimo objekto judėjimo kampą ir pagal jį parinkti atitinkamus animacijos kadrus.

Žaidėjo judėjimo kampas nustatomas pagal formulę:

kampas = *atan2*(x_1-x_2, y_1-y_2), kur x_1-x_2 yra pokytis tarp dabartinės ir prieš tai buvusios pozicijos x koordinačių ašyje, o y_1-y_2 yra pokytis tarp dabartinės ir prieš tai buvusios pozicijos y koordinačių ašyje.

Žinodami kampą, galime parinkti atitinkamą animaciją. Animacija nustatoma naudojant tokį algoritmą:

1. Pasirinktą kampą daliname iš kapo tarp animacijų skirtumo, t.y. 360/16=22.5.
2. Rezultatą apvalinime ir gauname atitinkamą animacijos skaitinę reikšmę iš skirtingų 16 animacijų masyvų.

Tarp kiekvieno kampo nustatymo reikalinga tiesinė kampo reikšmių interpoliacija tam, kad pavyzdžiui esant ties 300 laipsnių kampu ir žaidimo objektą pasukus ties 5 laipsnių kampu, objektas suktųsi ne atgal 295 laipsnius, o 65 laipsniais į priekį. Taip imituojamas natūralus objekto judėjimas. Taip pat interpoliacija padeda sumažinti objekto drebėjimą dėl triukšmingų akselerometro duomenų.

Tiesinę interpoliaciją galime apskaičiuoti pagal funkciją [19]:

$y = (x_1 + (x_2 - x_1) \cdot t)$, čia t - maišymo faktorius.

4.2 Sistemos architektūra – statinės struktūros modelis

4.2.1 Loginė visos sistemos architektūra

Akselerometru valdomo žaidimo sistemos loginė architektūra pavaizduota 4.8 pav. Ją sudaro 3 pagrindinės dalys:

Programų talpinimo posistemis

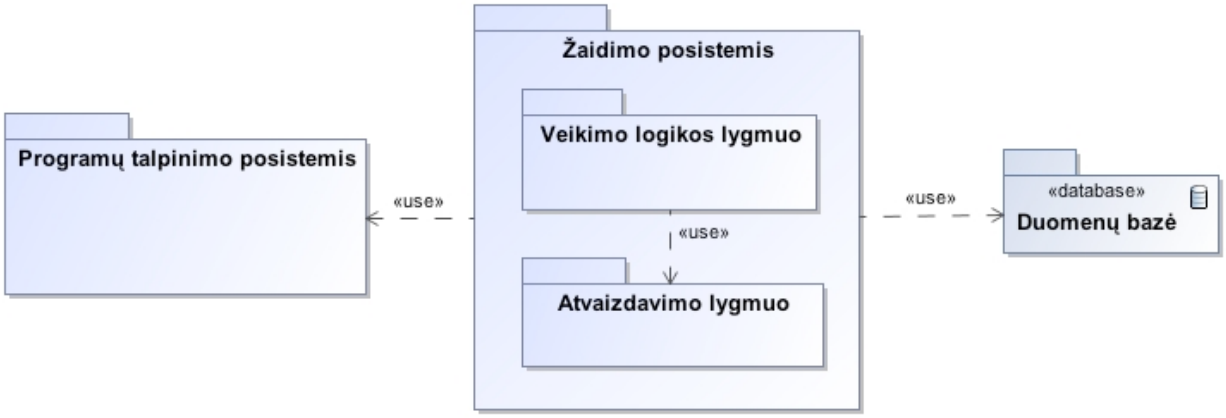
Šis posistemis apibrėžia AppStore programų talpinimo paslaugą. Čia talpinamas ir darbo metu realizuotas žaidimas.

Žaidimo posistemis

Šis posistemis yra pats svarbiausias. Jis išskirtas į "Veikimo logikos" ir "Atvaizdavimo" lygmenis. Čia vyksta pagrindiniai žaidimo procesai, logika ir atvaizdavimas.

Duomenų bazė

Šis posistemis yra duomenų paslaugų posistemis ir aprašo duomenų bazės struktūrą ir veiksmus su ja.



4.8 pav. Akselerometru valdomo žaidimo sistemos loginė architektūra

4.2.2 Vartotojo paslaugos

Vartotojo sąsaja

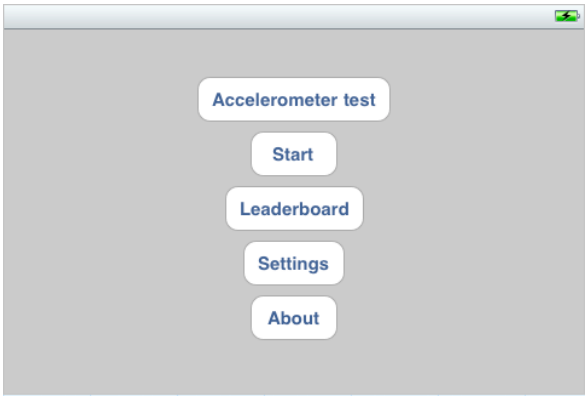
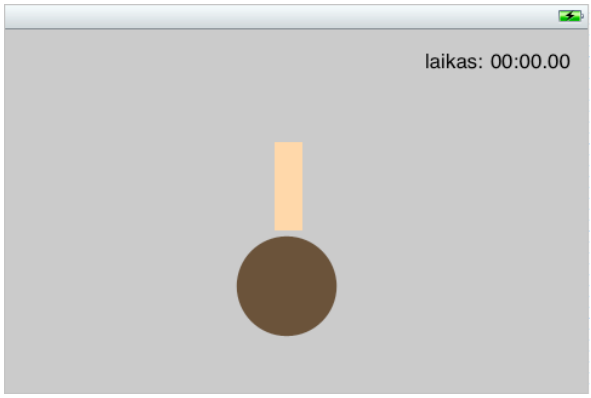
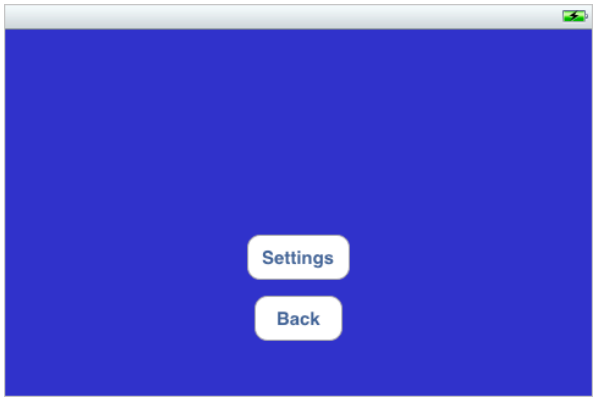
Sistemoje apibrėžti šie pagrindiniai vartotojo sąsajos langai:

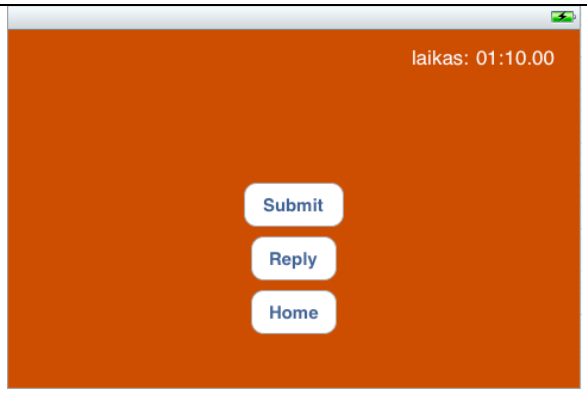
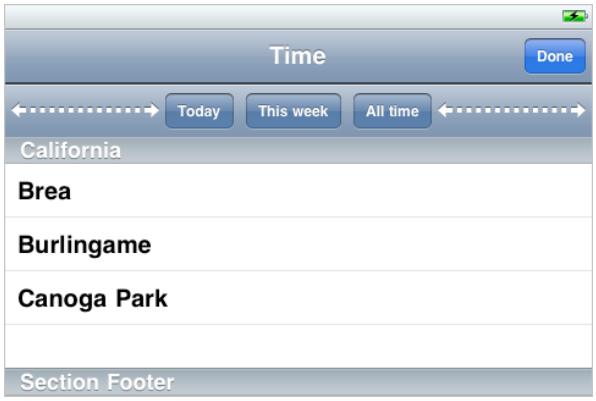
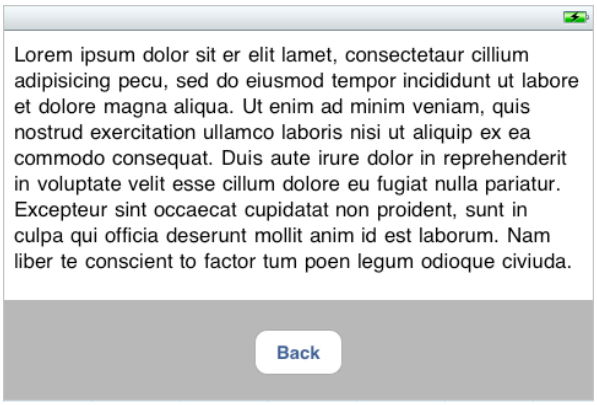
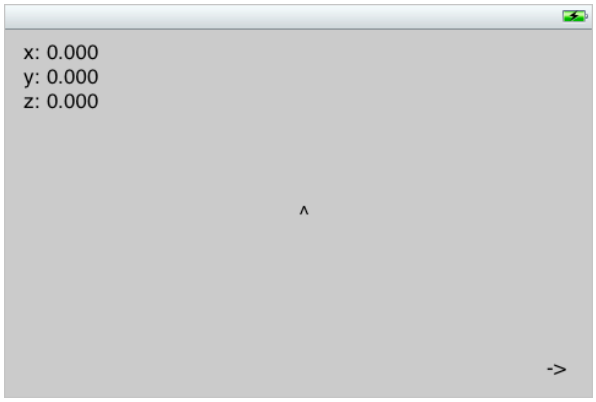
- Krovimosi langas,
- Pradinis langas,
- Akselerometro testavimo langas,
- Žaidimo langas, žaidimo sustabdymo langas, žaidimo pabaigos langas,
- Rekordų langas,

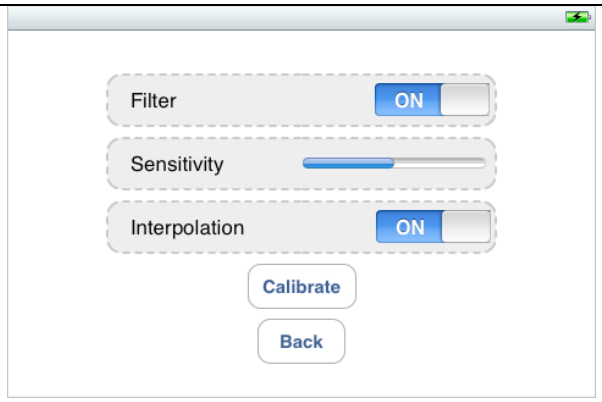
- Nustatymų langas,
- Apie langas.

4.1 lentelė. pateikti vartotojo sąsajos modeliai ir jų aprašymai.

4.1 lentelė. Vartotojo sąsajos langai

Langas	Aprašymas
Pradinis langas 	Pagrindiniame lange yra 5 mygtukai: • Testas, • Pradžia, • Nustatymai, • Rekordai, • Apie. Visi šie mygtukai atidaro naują atitinkamą langą.
Žaidimo langas 	Žaidimo lange yra žaidimo objektas ir žaidimo laikas. Žaidimas sustabdomas paspaudus ant žaidimo lango. Išviečiamas žaidimo sustabdymo langas.
Žaidimo sutabdymo langas 	Žaidimo sustabdymo lange yra 2 mygtukai: • Nustatymai, • Atgal. “Nustatymai” mygtukas atidaro nustatymų langą.
Žaidimo pabaigos langas	Žaidimo pabaigos lange yra 2 mygtukai: • Pateikti rezultatą,

	• Atgal. “Pateikti rezultatą” iškviečia rekordų langą.
Rekordų langas 	Rekordų lange yra vienintelis mygtukas: • Atgal. Rekordai pateikiami sąrašo tipo elemente, kurį galima persukti lietimo pagalba.
Apie langas 	Apie lange yra vienintelis mygtukas: • Atgal. Čia pateikiama autoriaus informacija.
Testavimo langas 	Testavimo lange matomas valdomas objektas, stiprumo vektorius, akseleracijos reikšmės. Paspaudus ant testavimo lango iškviečiamas pauzės langas, kuriame galima keisti tam tikrus nustatymus.
Nustatymų langas	Nustatymų lange pateikiami pagrindiniai metodo

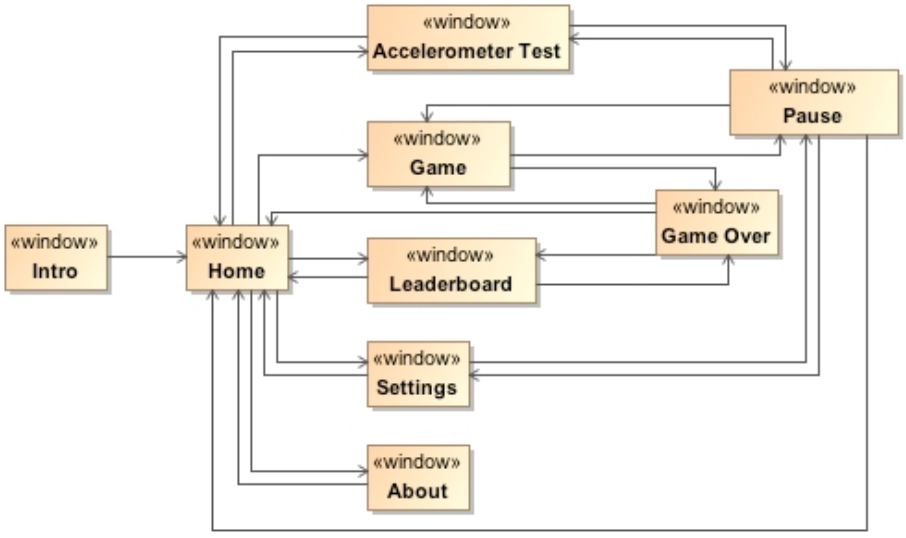


nustatymai:

- Filtras įjungtas/išjungtas,
- Intensyvumas,
- Interpoliacija įjungta/išjungta,
- Kalibravimas.

Vartotojo navigavimo planas

Vartotojo navigavimo planas pateiktas 4.9 pav. Navigavimo planas yra paprastas, nes žaidimo sistema susideda iš 4 pagrindinių langų. Visi sudėtingi procesai vyksta žaidimo lange. Vartotojas iš pagrindinio lango gali nueiti į Žaidimo, Rekordų ir Apie langus. Iš visų šių langų galima grįžti atgal į pagrindinį langą. Žaidimo lange yra galimybė atidaryti rekordų langą.



4.9 pav. Vartotojo navigavimo planas

4.2.3 Veiklos paslaugos

4.10 pav. pateikta žaidimo posistemio, veikimo logikos lygmens veiklos klasių diagrama.

Veikimo logikos lygmens veiklos klasių diagrama susideda iš pagrindinio žaidimo lango, kurį su žaidimo objektu ir jo aplinka susieja žaidimo valdiklis.

4.10 pav. Žaidimo posistemio, veikimo logikos lygmens veiklos klasių diagrama

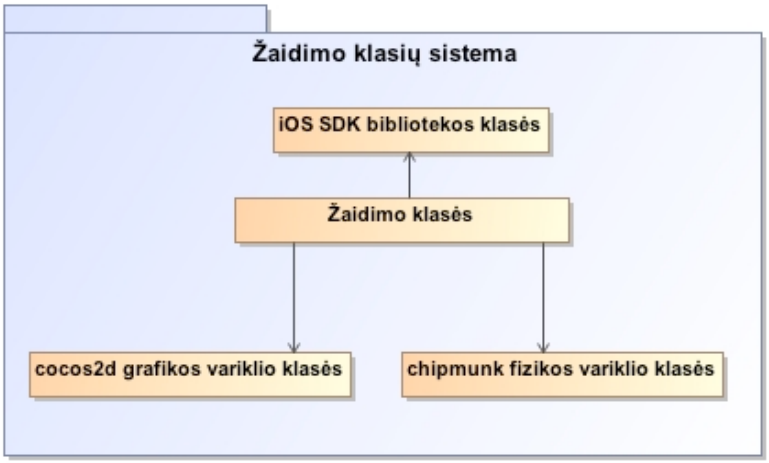
4.11 pav. pateikta žaidimo posistemio, atvaizdavimo lygmens veiklos klasių diagrama. Žaidimo atvaizdavimo veiklos klasių diagrama susideda iš pagrindinio žaidimo lango, kurį su žaidimo objektu ir jo aplinka susieja atvaizdavimo valdiklis.

4.11 pav. Žaidimo posistemio, atvaizdavimo lygmens veiklos klasių diagrama

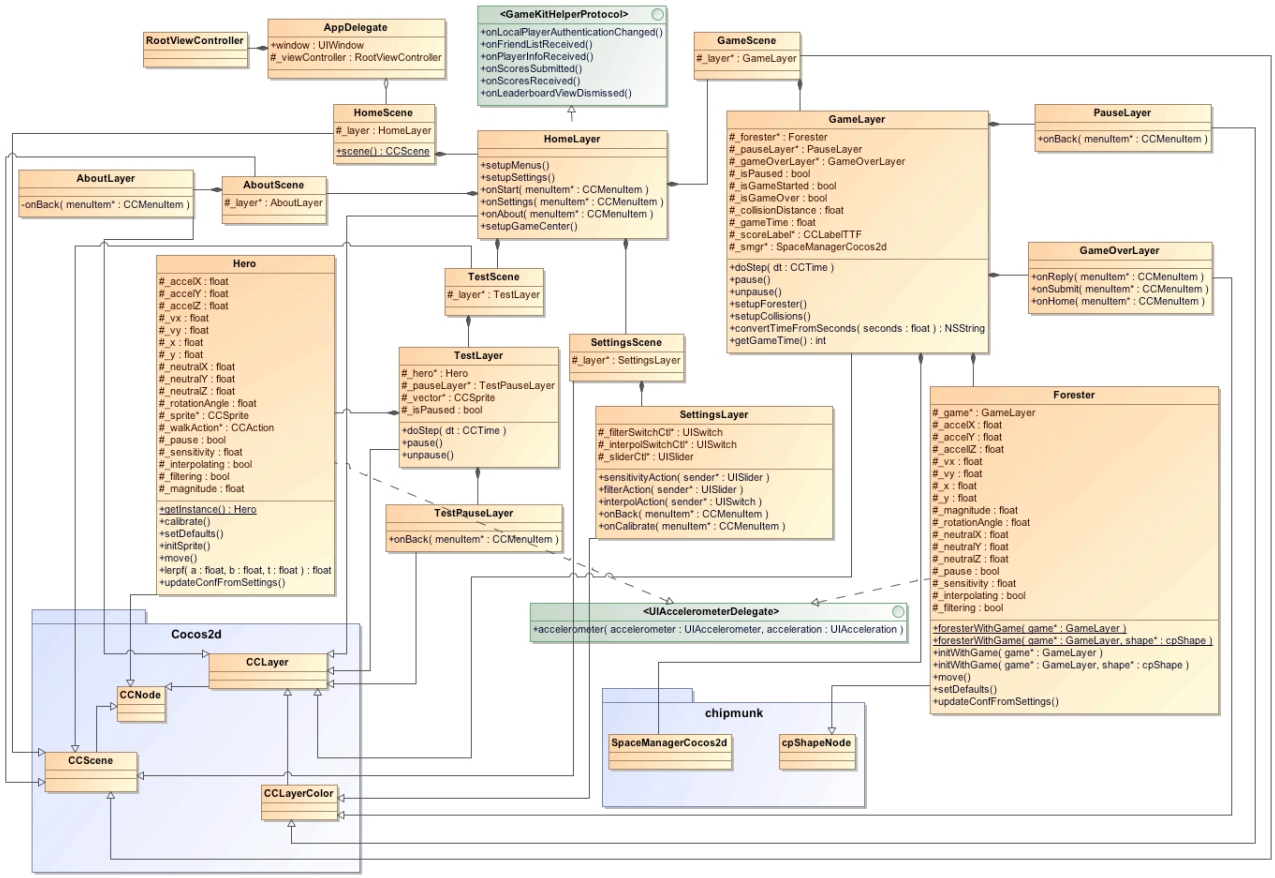
4.3 Detalus projektas

Žaidimo klasių sistemą (4.12 pav.) sudaro iOS SDK pateikiama klasių biblioteka, kuri padeda valdyti techninę prietaiso įrangą ir turi priemones, leidžiančias reaguoti į prietaiso pasukimus, ekrano lietimus, valdyti garso išvedimą. Kiekviena priemonė sudaro didžiulę biblioteką ir čia nebus detaliai aprašyta.

Be pagalbinių iOS SDK priemonių, jos turi būti naudojamos jau programuotojo aprašytuose klasėse, o taip pat reikalinga tam tikra žaidimo logika, žaidimo elementai, aplinka ir aplinkos objektai, kurie sudaro visą žaidimo sistemos visumą. Pagrindinės klasės, susiejančios iOS SDK, žaidimą ir jo atvaizdavimą pateiktos 4.13 pav. Klasių paskirtis trumpai aprašyta 4.2 lentelė.



4.12 pav. Naudojamų klasių sistema.



4.13 pav. Žaidimo klasių diagrama

4.2 lentelė. Klasių aprašymas

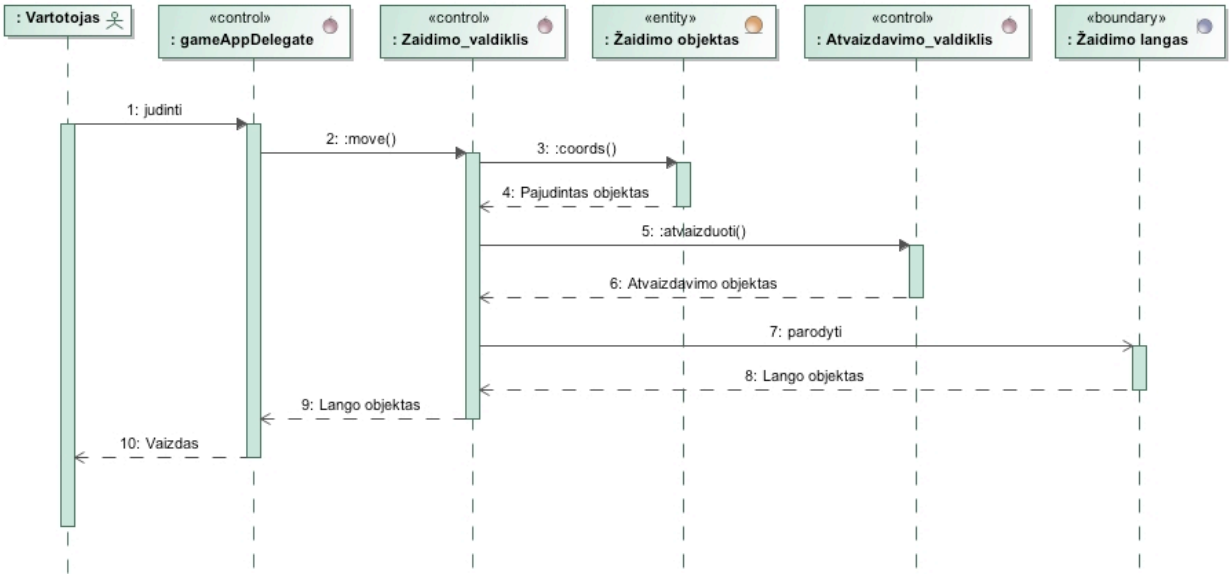
Klasė	Aprašymas
RootViewController	Valdo ekrano vaizdo padėtį, inicijuoja pradinius vaizdo pasukimo parametrus.
AppDelegate	Pagrindinė programos klasė, kuri aptarnauja veikiančią programą. Paleidimo metu ji inicijuoja programos pradinius parametrus, nustato vaizdo parametrus, inicijuoja grafikos varikliuką ir galiausiai iškviečia pirmą programos langą. Programos išjungimo metu pasirūpina atminties valymu.
HomeScene, HomeLayer	Pradinė žaidimo paleidimo scena ir sluoksnis. Aprašo pradinį meniu, inicijuoja prisijungimą prie nutolusio serverio rekordams fiksuoti.
TestScene, TestLayer	Akselerometro testavimo scena ir sluoksnis. Ši scena skirta akselerometro galimybėms pavaizduoti ir eksperimentiniams rezultatams sekti.
TestPauseLayer	Papildomas sluoksnis, skirtas sustabdyti testavimo aplinką, iškviesti nustatymo sluoksnį arba grįžti į pagrindinę sceną.
GameScene, GameLayer	Žaidimo scena ir sluoksnis, kuriame realizuotas akselerometru valdomas žaidimas, remiantis sukurtu valdymo metodu.
PauseLayer	Žaidimo sustabdymo sluoksnis grįžimui į pagrindinį meniu.
GameOverLayer	Žaidimo pabaigos sluoksnis, kuriame galima pateikti pasiektą rezultatą į rezultatų debesį, kartoti žaidimą arba visai jį nutraukti.
SettingsScene, SettingsLayer	Nustatymų scena ir sluoksnis skirti atlikti tam tikrus nustatymų pakeitimus: pakeisti jautrumą, filtravimą, interpoliaciją bei atlikti akselerometro kalibravimą.

AboutScene, AboutLayer	Apie scena ir sluoksnis skirti informacijai apie kūrėją pateikti.
Hero	Testinės scenos valdymo objektas, reaguojantis į akselerometro pokyčius ir valdomas sukurtu valdymo metodu.
Forester	Žaidimo valdymo objektas, reaguojantis į akselerometro pokyčius ir valdomas sukurtu valdymo metodu.
GameKitHelperProtocol	Žaidimo rekordų pagalbinis interfeisas, padedantis prisijungti prie nutolusio duomenų centro ir saugoti rekordų reikšmes.
UIAccelerometerDelegate	Akselerometro delegato interfeisas, perduodantis akseleracijos reikšmę delegato objektui.
CCNode	Grafikos variklio pagrindinė klasė, skirta piešiamų objektų medžio funkcionalumui imituoti. Pagrindinės savybės: gali turėti kitus CCNode objektus, turi laikmatį, turi veiksmų funkcionalumą (judėjimas, sukimas ir t.t.)
CCScene	Scenos klasė, kuri paveldi CCNode klasę. Scena skirta vienam ekrane matomam vaizdui atvaizduoti. Scena susideda iš vieno ir daugiau sluoksnių.
CCLayer	Sluoksnis, naudojamas skirtingam funkcionalumui ekrane pateikti. Pvz. meniu ir tam tikra animacija yra skirtinguose sluoksniuose. Kiekvienam sluoksniui galima taikyti skirtingas judesio taisykles.
CCLayerColor	Sluoksnis, turintis spalvą.
SpaceManagerCocos2d	Pagalbinė klasė, skirta fizikos dėsniam generuoti: sukurti erdvę, gravitacijos jėgas, masės kūnus ir jų tarpusavio sąryšius.
cpShapeNode	Masės kūno klasė, turinti svorį, masės centrą, pagreitį.

4.4 Sistemos elgsenos modelis

Žaidimo valdymo sekų diagrama pavaizduota 4.14 pav.

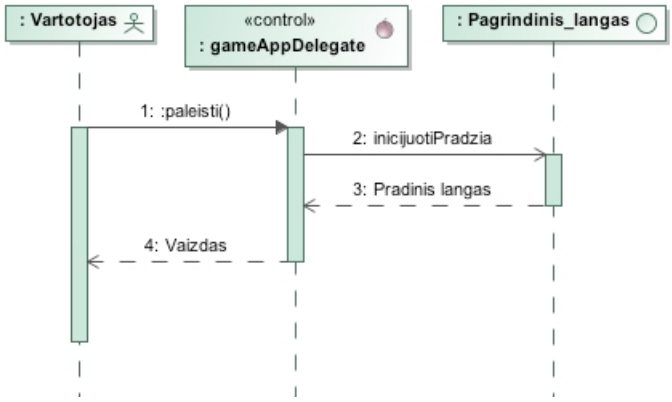
Vartotojas pajudina prietaisą. Judinimo duomenis iš UIAccelerometer objekto gauna gameAppDelegate objektas ir juos perduoda Zaidimo_valdiklis objektui, o šis reikiama kryptimi perkelia žaidimo objektą ir perduoda naują vaizdą į Atvaizdavimo_valdiklis objektą. Čia sukuriamas naujas vaizdas žaidimo lange kurį mato vartotojas.



4.14 pav. Žaidimo valdymo sekų diagrama, atitinkanti PA "Valdyti žaidimą"

Žaidimo paleidimo sekų diagrama pateikta 4.15 pav.

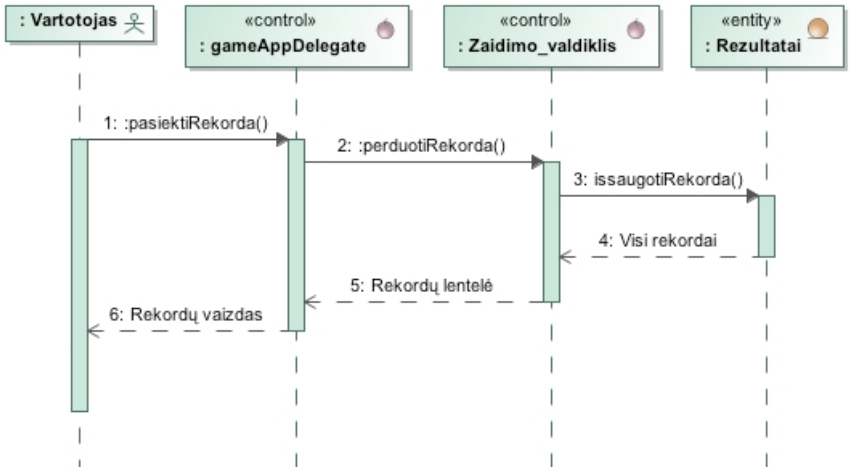
Vartotojas paleidžia žaidimą, o gameAppDelegate objektas inicijuoja žaidimo pradžią ir vartotojui parodomas pradinis žaidimo langas.



4.15 pav. Žaidimo paleidimo sekų diagrama, atitinkanti PA "Paleisti žaidimą"

Rekordų įrašymo sekų diagrama pateikta 4.16 pav.

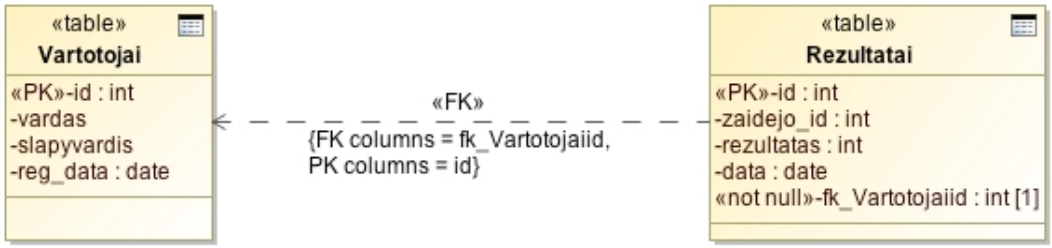
Vartotojas pasiekia rekordą, jį perduoda Zaidimo_valdiklis objektui, kuris saugo duomenis nutolusioje duomenų bazėje. Tada vartotojui grąžinamas rekordų lentelės vaizdas.



4.16 pav. Rekordų įrašymo sekų diagrama atitinkanti PA "Pateikti rekordus"

4.5 Duomenų bazės schema

Rekordų duomenų bazės schema pateikta 4.17 pav. Rekordų duomenų bazę sudaro 2 lentelės, aprašančios prisiregistravusį vartotoją ir jo pasiektą rekordą tam tikru metu. Kiekvienas vartotojas gali saugoti neribotą kiekį rezultatų. Vartotojas, norėdamas pateikti tam tikrą rekordą, privalo būti užsiregistravęs su tam tikru vartotojo vardu ir norimu slaptažodžiu. Lentelėje yra saugoma vartotojo registracijos data, kuri padės nustatyti bendrą vartotojų didėjimo arba mažėjimo tendenciją per tam tikrą laikotarpį.



4.17 pav. Rekordų duomenų bazės schema

4.6 Realizacijos modelis

Akselerometru valdomo žaidimo komponentų modelis pateiktas 4.18 pav. Komponentų modelyje pateikiami sistemos komponentai, aprašantys sudedamąsias sistemos dalis ir sąveika tarp jų.

Modelį sudaro tokios dalys:

iOS SDK

Šis komponentas susideda iš 2 sudėtinių dalių:

- Touch UI – lietimui jautrios vartotojo sąsajos komponentas.
- MVC – modelis-vaizdas-valdiklis projektavimo modelio komponentas.

Žaidimas

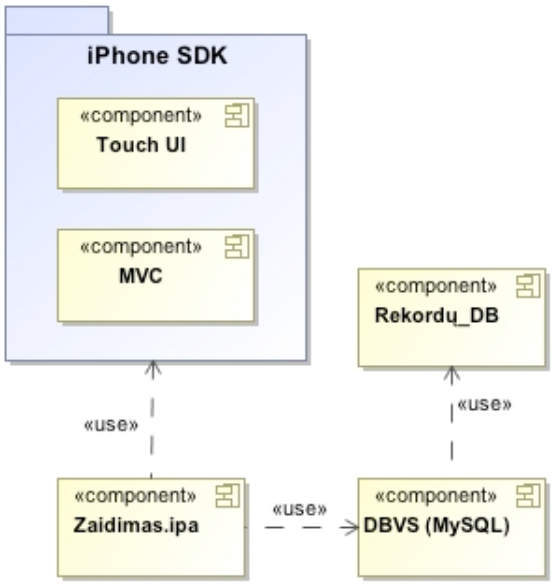
Tai visa žaidimo sistema, paleidžiamasis jos binarinis sukompiliuotas failas, kuris automatiškai įrašomas į įrenginį iš AppStore programų saugojimo sistemos.

Duomenų bazių valdymo sistema

Sistema, naudojama vartotojų ir rekordų duomenims saugoti ir valdyti.

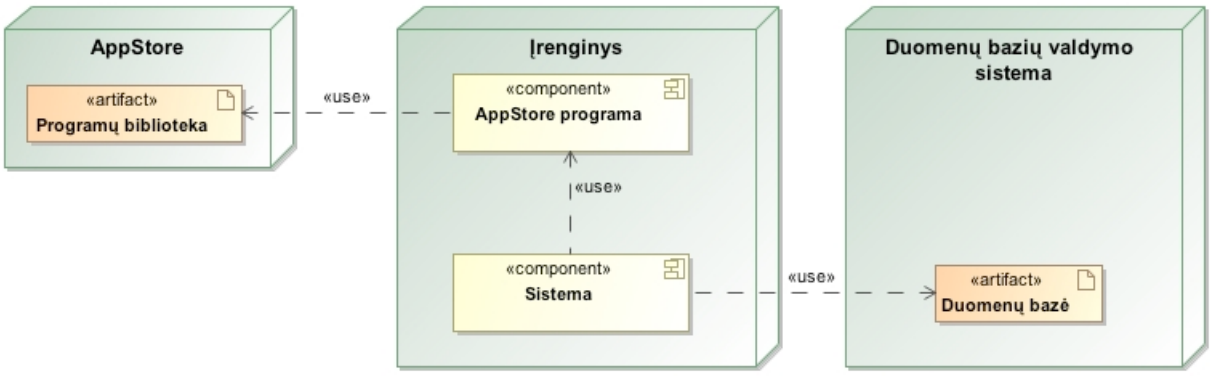
Rekordų duomenų bazė

Komponentas, aprašantis visus žaidėjų rekordus tam tikru momentu.



4.18 pav. Akselerometru valdomo žaidimo sistemos komponentų modelis

Akselerometru valdomo žaidimo diegimo modelis pateiktas 4.19 pav. Diegimas pradedamas nuo programos parsisiuntimo iš AppStore programų saugojimo sistemos. Programa automatiškai įdiegiama į įrenginio operacinę iOS sistemą. Ši sistema sąveikauja su nutolusia rekordų duomenų baze (Apple debesiu).



4.19 pav. Akselerometru valdomo žaidimo diegimo modelis

5 Realizacija

Pirmoje dalyje bus aptarta sukurto metodo realizacija, pateiktas sprendimo aprašymas, o antroje dalyje – žaidimo prototipo realizacija pritaikius sukurtą metodą. Taigi, realizaciją galima padalinti į dvi dalis:

- sukurto metodo realizacija,
- žaidimo prototipo realizacija, panaudojant sukurtą metodą.

5.1 Sistemos veikimo aprašymas

5.1.1 Sukurto metodo realizacija

Sukurtam metodui testuoti ir stebėti tam tikrus koeficientus buvo realizuota speciali testavimo aplinka, pavaizduota 5.1 pav. Šios testavimo aplinkos pagrindiniai komponentai:

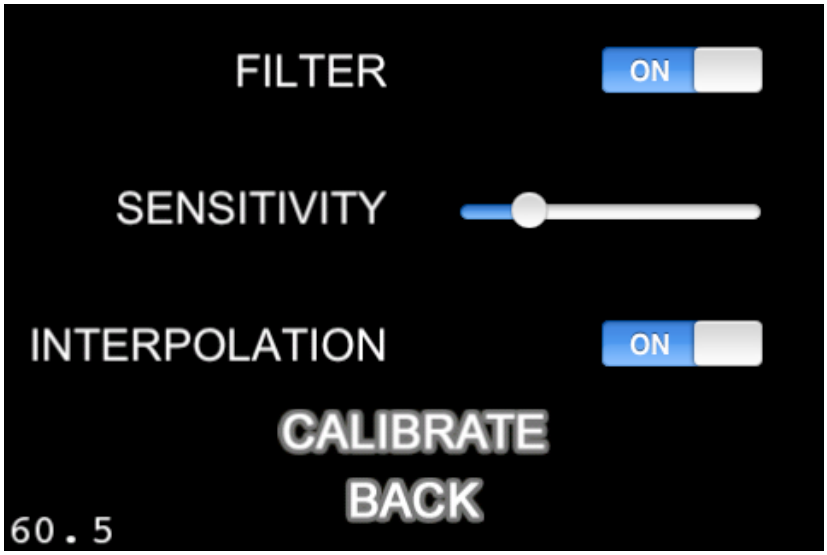
- akselerometru valdomas objektas (baltas trikampis), kurio smailioji pusė nurodo judėjimo kryptį,
- esamuoju laiku pateikiamos akselerometro reikšmės x, y ir z ašyse, padedančios stebėti įvairių filtrų, jautrumo, interpoliacijos įtaką akselerometro duomenims,
- judėjimo stiprumo-krypties vektorius (mėlyna rodyklė), nurodantis ir judėjimo kryptį, ir judesio stiprumą, priklausantį nuo prietaiso laikomos pozicijos.

5.1 pav. Akselerometro testavimo aplinka

Testavimo aplinkoje taip pat galimi tam tikri nustatymai, leidžiantis tiesiog testavimo aplinkoje keisti įvairius akselerometro duomenis įtakojančius parametrus, pavaizduoti 5.2 pav.:

- galimybė įjungti/išjungti duomenų filtravimą,
- galimybė nustatyti akselerometro judėjimo jautrumą,
- galimybė įjungti/išjungti posūkio kampo interpoliaciją,

- galimybė rankiniu būdu atlikti akselerometro kalibravimą.



5.2 pav. Akselerometro nustatymų langas

Valdymo metodas

Pagrindinis algoritmo žingsnis, nustatantis galutinę judėjimo kryptį. Metodo veikimo idėja aprašyta 4.1 skyriuje. Žaidimo objekto poslinkio x ir y ašimis nustatymo algoritmo programos fragmentas:

```
// akselerometro delegato metodas
   - (void) accelerometer:(UIAccelerometer*)accelerometer
      didAccelerate:(UIAcceleration*)acceleration
{
    // judėjimo vektoriaus projekcijos vektorius
    b2Vec2 accel2d = b2Vec2(0, 0);

    b2Vec3 ax = b2Vec3(1, 0, 0);
    b2Vec3 ay = b2Vec3(-0.63f, 0, -0.92f); // pradinis vektorius

    b2Vec3 az = b2Cross(ax, ay).Normalize(); // ekrano vektorius y
    ax = b2Cross(az, ay).Normalize();// ekrano vektorius x

    accel2d.x = -b2Dot(b2Vec3(acceleration.x, acceleration.y,
    acceleration.z), ax); // judėjimo vektoriaus projekcija x ašyje
    accel2d.y = -b2Dot(b2Vec3(acceleration.x, acceleration.y,
    acceleration.z), az); // judėjimo vektoriaus projekcija y ašyje

    const float xSensitivity = self.sensitivity;
    const float ySensitivity = self.sensitivity; // akselerometro
    intensyvumas
    const float tiltAmplifier = 8; // pakreipimo stiprintuvas

    vx += (accel2d.y) * tiltAmplifier * xSensitivity; // x poslinkis

    vy += (accel2d.x) * tiltAmplifier * ySensitivity; // y poslinkis
}
```

Triukšmingų duomenų filtravimas

Žemų dažnių duomenų filtravimo algoritmo realizacijos fragmentas:

```
-(void)lowPassFilter
{
    self.x = self.accelX * kLowPassFilteringFactor + self.x * (1.0 -
kLowPassFilteringFactor);
    self.y = self.accelY * kLowPassFilteringFactor + self.y * (1.0 -
kLowPassFilteringFactor);
    self.z = self.accelZ * kLowPassFilteringFactor + self.z * (1.0 -
kLowPassFilteringFactor);
}
```

Čia *self.accelX, self.accelY, self.accelZ* – dabartinės x, y, z ašių akseleracijos reikšmės, *kLowPassFilteringFactor* – išlyginimo faktoriaus konstanta, *self.x, self.y, self.z* – dabartinės x, y, z ašių reikšmės

Aukšų dažnių duomenų filtravimo algoritmo realizacijos fragmentas:

```
-(void)highPassFilter
{
    self.x = kLowPassFilteringFactor * (self.x + self.accelX -
self.lastX);
    self.y = kLowPassFilteringFactor * (self.y + self.accelY -
self.lastY);
    self.z = kLowPassFilteringFactor * (self.z + self.accelZ -
self.lastZ);
}
```

Realizuojami abu filtravimo algoritmai, o kuris geriau tinka šiam metodui bus patikrinta eksperimento metu.

Žaidimo objekto pasukimo kampas ir animacija

Pasukimo kampo problemos aprašomos 4.1 skyriuje. Žaidimo objekto pasukimo kampo nustatymo algoritmo fragmentas:

```
-(void)move
{
    float lastX = self.x;
    float lastY = self.y;

    const float deadZone = 0.2f;
    // jei įjungtas filtravimas - filtruojame
    if(self.filtering){
        [self lowPassFilter];
    }

    float diffx = self.x - lastX; // x skirtumas
    float diffy = self.y - lastY; // y skirtumas
    // jei nėra didelio drebėjimo, vykdom toliau
    if(b2Distance(b2Vec2(0, 0), b2Vec2(diffx, diffy)) >= deadZone){
     float desiredAngle = atan2(diffy, diffx); // norimas kampas
     float currentAngle = self.rotationAngle; // esantis kampas
```

```
      const float blendFactor = 0.35f; // maišymo faktorius
         // ar naudojam interpoliaciją?
         if(self.interpolating){
             self.rotationAngle = Slerp2D(currentAngle, desiredAngle,
blendFactor);
         } else {
             self.rotationAngle = desiredAngle;
         }

         // konvertuojam į laipsnius
         float degrees = -1 * CC_RADIANS_TO_DEGREES(self.rotationAngle);
         if(degrees < 0){
             degrees = degrees + 360;
         }
         // nustatom pasukimą
         _sprite.rotation = degrees;
     }
}
```

5.1.2 Žaidimo prototipo realizacija, panaudojant sukurtą metodą

Žaidimo scena

Pagrindinė žaidimo idėja – akselerometro pagalba kuo ilgiau nenukritus balansuoti žaidimo objektą ant besisukančio susidūrimo objekto (apskritimo). Pasiekus tam tikrą kritimo ribą, žaidimas sustabdomas ir siūloma savo geriausią rezultatą patekti į bendrą rekordų duomenų bazę.

Žaidimas paremtas sukurtu valdymo metodu, kuris akselerometro pagalba leidžia žaidimo objektą balansuoti ant besisukančio rato.

Žaidime taip pat naudojamas fizikos variklis, kuris visus objektus paverčia fizikiniais kūnais, turinčiais masės centrą, masę ir yra veikiami gravitacijos jėgų. Valdomas objektas, užkritęs ant apskritimo ir veikiamas gravitacijos jėgų, pradeda kristi į vieną ar kitą pusę. Žaidėjas akselerometro pagalba įtakoja valdomo objekto lygsvarą dažniausiai prietaisą pasukdamas į priešingą pusę nuo kritimo.

Pagrindiniai žaidimo komponentai pavaizduoti 5.3 pav. Žaidimas susideda iš šių komponentų:

1. Akselerometru valdomas žaidimo objektas, turintis fizinę masę.
2. Besisukantis ant ašies susidūrimo objektas, turintis fizinę masę.
3. Žaidimo laikas, matuojantis balansavimo trukmę.
4. Prisijungimo langas/pranešimas apie sėkmingą prisijungimą.

5.3 pav. Žaidimo scena

Pagrindinė žaidimo objekto valdymo idėja – akselerometro pagalba keisti gravitacijos vektoriaus jėgas x ašyje. Tokiu būdu įtaka daroma ne tiesiogiai valdymo objektui, o visai erdvei, kurioje galioja įvairūs fizikos dėsniai. Kitaip valdymo realizuoti neįmanoma dėl fizikos varikliuko ypatybių. Toliau pateikiamas žaidimo objekto valdymo algoritmas, paremtas sukurtu valdymo metodu:

```
// aukščiau inicijuojamas paveldėtas valdymo metodas, nustatantis
akselerometro reikšmes
-(void) move
{
    // keičiame gravitacijos vektoriaus jėgas x ašyje
    [_game smgr].gravity = cpv(self.vx*kGravityMagnitude, -
9.8*kGravityMagnitude);
}
```

kGravityMagnitude - gravitacijos stiprintuvo koeficientas, apskaičiuotas bandymų metu. *self.vx* – anksčiau aprašytu valdymo metodu gautas poslinkis x ašimi. *9.8* – laisvojo kritimo pagreitis.

Momentas, kada žaidimo objektas kritimo metu pasiekia tam tikrą ribą, kai atstumas tarp jo ir besisukančio apskritimo yra didesnis už bandymų metu nustatytą koeficientą, yra vadinamas žaidimo pabaiga. Tokiu atveju fiksuojamas galutinis žaidimo laikas 5.4 pav.

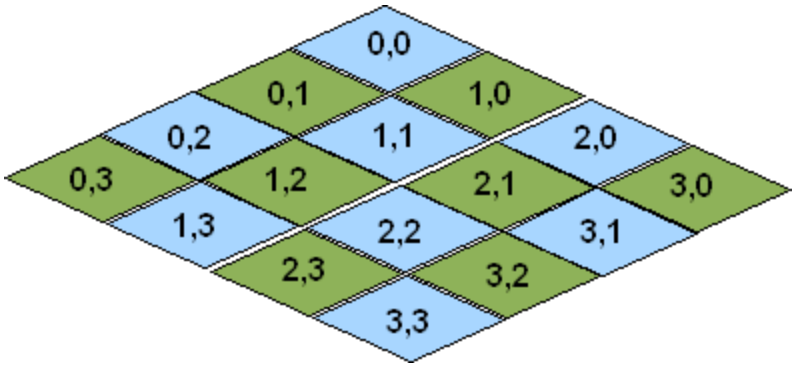
5.4 pav. Žaidimo pabaigos scena

Jei žaidėjas yra prisijungęs prie savo paskyros, jis gali pateikti rezultatą į bendrą duomenų bazę.

Plačiau apie žaidimo valdymą, rezultato pateikimą, nustatymus aprašyta priedų 10.1 skyriuje.

Žaidimo objekto susidūrimų su kitais objektais nustatymas

Žaidimo objektas juda izometriniame žemėlapyje. Vienas iš svarbiausių uždavinių prieš atliekant susidūrimų su kitais objektais nustatymą, nustatyti judinamo objekto poziciją šiame žemėlapyje. Izometrinio žemėlapio koordinačių sistema pateikta 5.5 pav. Pirmoji reikšmė rodo x koordinatę, antroji – y. Reikšmės pradedamos skaičiuoti nuo nulinės reikšmės, taigi paveiksle parodytas žemėlapis susideda iš 4 blokų. [4]



5.5 pav. Izometrinio žemėlapio koordinačių sistema

Toliau pateikiamas algoritmo fragmentas, grąžinantis objekto poziciją izometriniame žemėlapyje.

```
-(CGPoint) tilePosFromLocation:(CGPoint)location
tileMap:(CCTMXTiledMap*)tileMap
 {
     CGPoint pos = ccpSub(location, tileMap.position);
```

```
      float halfMapWidth = tileMap.mapSize.width * 0.5f;
      float mapHeight = tileMap.tileSize.height;
      float tileWidth = tileMap.tileSize.width;
      float tileHeight = tileMap.tileSize.height;

      CGPoint tilePosDiv = CGPointMake(pos.x / tileWidth, pos.y /
tileHeight);
      float inverseTileY = mapHeight - tilePosDiv.y;

      float posX = (int)(inverseTileY + tilePosDiv.x - halfMapWidth);
      float posY = (int)(inverseTileY - tilePosDiv.x + halfMapWidth);

      posX = MAX(0, posX);
      posX = MIN(tileMap.mapSize.width - 1, posX);
      posY = MAX(0, posY);
      posY = MIN(tileMap.mapSize.width - 1, posY);

      return CGPointMake(posX, posY);
 }
```

Žaidimo objekto susidūrimas su kitais objektais nustatomas atliekant žaidimo objekto pozicijos palyginimą su kito objekto pozicija izometrinio žemėlapio koordinačių sistemoje. Šiuo atveju izometrinis žemėlapis turi atitinkamą susidūrimų sluoksnį, kuriame saugomi susidūrimo objektai su tam tikru parametru "*blocks_movement*". Susidūrimo tikrinimo funkcija atrodo taip:

```
-(bool) isTileBlocked:(CGPoint)tilePos tileMap:(CCTMXTiledMap*)tileMap
{
    CCTMXLayer *collisionsLayer = [tileMap layerNamed:@"Collisions"];
    bool isBLocked = NO; // nėra susidūrimo, pradinė reikšmė
    unsigned int tileGID = [collisionsLayer tileGIDAt:tilePos];
    if (tileGID > 0) { // egzistuoja susidūrimo objektas
     NSDictionary *tileProperties = [tileMap propertiesForGID:tileGID];
     id blocks_movement = [tileProperties
objectForKey:@"blocks_movement"]; // egzistuoja susidūrimo parametras
      isBLocked = (blocks_movement != nil);
    }
    return isBLocked;
}
```

Kiekvieno kadro metu yra atliekamas susidūrimo tikrinimas. Pirmą kartą aptikus susidūrimą, inicijuojamas žaidimo pradžios laikas. Kai randamas atstumas tarp susidūrusių objektų yra didesnis už numatytą koeficientą, žaidimo laikas sustabdomas ir parodoma žaidimo pabaigos scena 5.4 pav.

5.2 Sukurto metodo ir jo realizacijos apibendrinimas

Realizacijos metu buvo praktiškai realizuotas projekto specifikacijoje apibrėžtas žaidimo valdymo metodas, paremtas kintančiais akselerometro duomenimis. Sukurtas metodas parašytas Objective-C programavimo kalba ir skirtas iOS operacinę sistemą palaikantiems įrenginiams. Šis

metodas pritaikytas žaidimo prototipo kūrime, kuris parodytų praktinę metodo pritaikymo pusę ir jo galimybes.

Sukurto metodo pagrindinės savybės:

- žaidimo objekto valdymas bet kokia kryptimi, paprastų gestų palaikymas,
- sklandus žaidimo objekto valdymas dėl duomenų filtravimo, pasukimo kampo efektyvaus interpoliavimo,
- akselerometro kalibravimo galimybė leidžianti vartotojui pačiam nusistatyti patogiausią valdymo padėtį,
- greičio ir stiprumo vektorių parodymai.

Sukurto žaidimo prototipo pagrindinės savybės:

- realizuotas remiantis sukurtu valdymo metodu,
- naudojamas fizikos varikliukas, leidžiantis žaidimo objektams tarpusavyje sąveikauti pagal fizikos dėsnius: gravitacijos laukas, masės centras, masė, kūnų sujungimai, susidūrimų kontrolė.
- Naudojama bendra Apple vartotojo paskyra prisijungimui prie žaidimo sistemos ir geriausių rezultatų publikavimui Apple rekordų duomenų debesyje.

6 Eksperimentinis sistemos tyrimas

Darbo metu buvo sukurtas naujas metodas, leidžiantis akselerometro duomenis efektyviai pritaikyti žaidimo objekto valdymui. Kadangi akselerometro duomenys, ateinantys tiesiai iš techninės įrangos, nėra tinkami tiesiogiai naudoti valdymui, juos būtina apdoroti tam tikrais metodais. Siekdami patikrinti sukurto metodo korektišką veikimą, jo galutinę vertę tarp nagrinėtų panašių sistemų bus atliekamas sukurto metodo nuodugnus testavimas:

- patikrintas skirtingų filtravimo algoritmų korektiškumas ir pasirinkti geriausi koeficientai,
- įvertintas "Gimbal lock" problemos sprendimas,
- patikrinta akselerometro jautrumo įtaka judėjimui,
- patikrinta judėjimo stiprumo priklausomybė nuo įrenginio pakreipimo kampo.

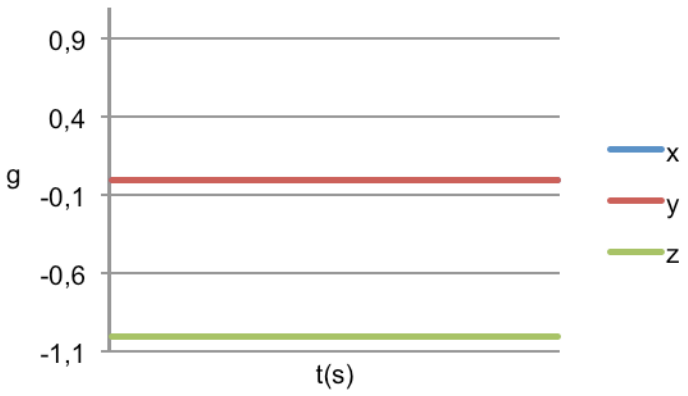
6.1 Savybių analizė

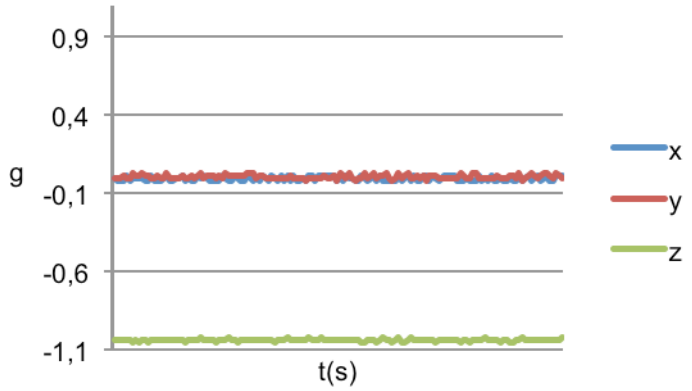
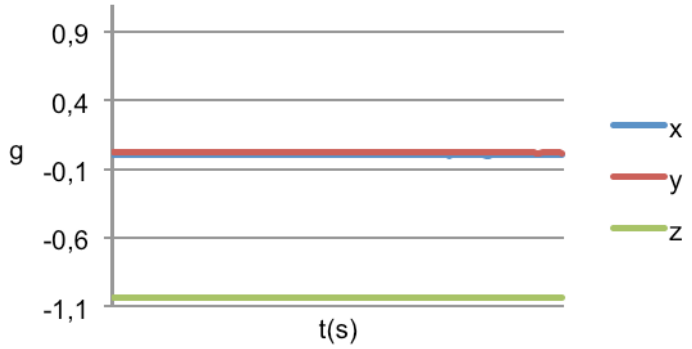
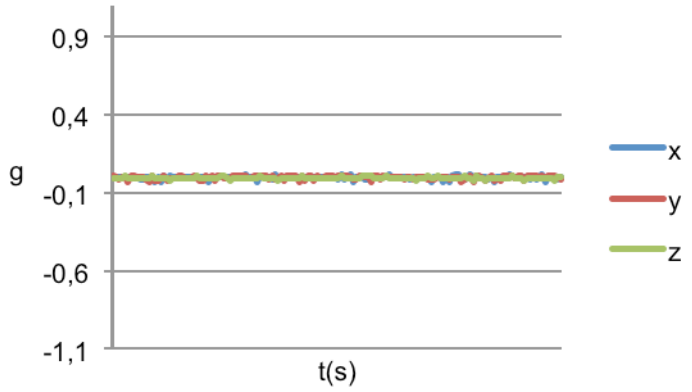
Duomenų filtravimas

Metodas buvo realizuotas panaudojant 2 skirtingus algoritmus akselerometro duomenims filtruoti: žemų dažnių filtras ir aukštų dažnių filtras. Matavimas atliktas fiksuojant akseleracijos duomenis tam tikrą laiką ir saugant juos CSV formatu.

6.1 lentelė. pavaizduoti akselerometro rodmenys pritaikius visus filtrus, kada įrenginys padėtas ramybės būsenoje ant stalo. Y ašyje nurodytas akseleracijos dydis, o X ašyje – laiko tarpas. Pirmame grafike pavaizduotas siekiamas rezultatas.

6.1 lentelė. Akselerometro rodmenys ramybės būsenoje ant stalo

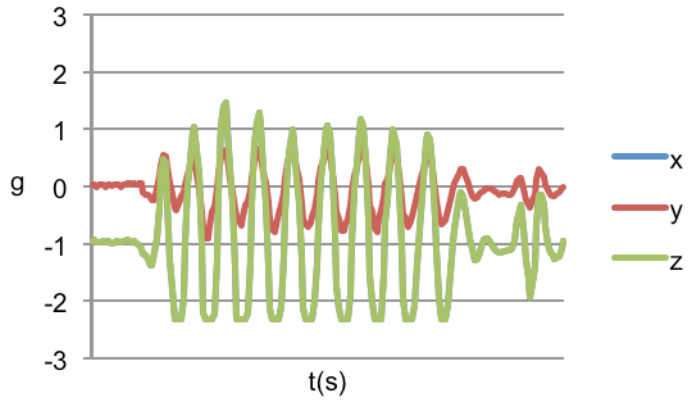
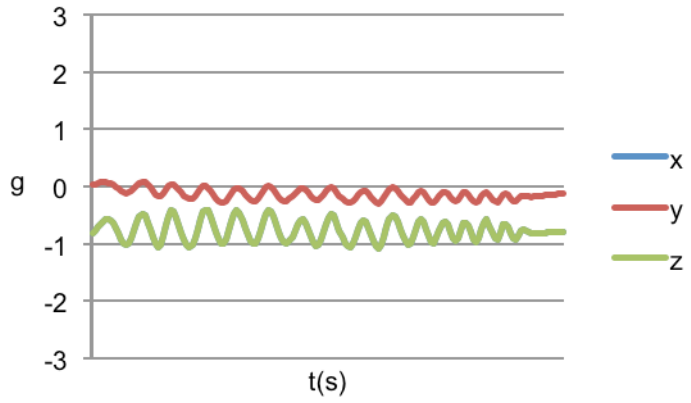
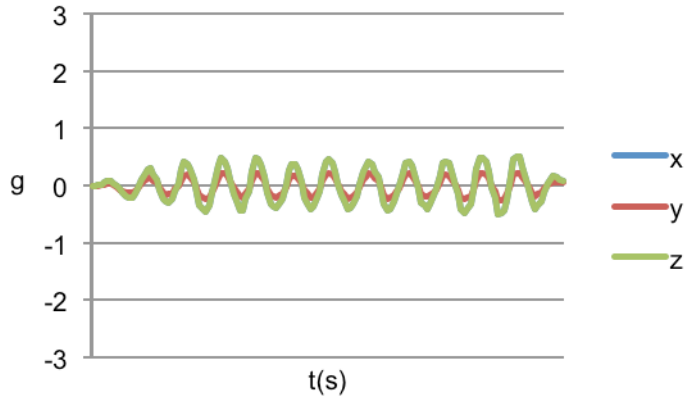
Filtro tipas	**Grafikas**
Idealūs ir siekiami akselerometro duomenys šioje pozicijoje	

Akselerometro rodmenys nenaudojant jokio filtro	
Akselerometro rodmenys pritaikius žemų dažnių filtrą	
Akselerometro rodmenys pritaikius aukštų dažnių filtrą	

6.2 lentelė. pavaizduoti akselerometro rodmenys pritaikius visus filtrus, kada įrenginys stipriai judinamas aukštyn-žemyn. Y ašyje nurodytas akseleracijos dydis, o X ašyje – laiko tarpas.

6.2 lentelė. Akselerometro rodmenys judinant aukštyn-žemyn

Filtro tipas	**Grafikas**

Akselerometro rodmenys nenaudojant jokio filtro	
Akselerometro rodmenys pritaikius žemų dažnių filtrą	
Akselerometro rodmenys pritaikius aukštų dažnių filtrą	

Kaip matome iš pastarųjų lentelių, net ir ramybės būsenoje, pastebimas didžiulis duomenų triukšmas. Pritaikę žemų dažnių filtrą pastebimai sumažiname triukšmą. Pastebėtina, kad ramybės būsenoje didžiausia akseleracija vyksta z ašyje. Matome, kad praleidžiamos tik žemų dažnių reikšmės. Tokius duomenis jau galima naudoti tolesniam valdymo programavimui. Pritaikę aukštų dažnių filtrą taip pat matome triukšmo sumažėjimą. Aukštų dažnių filtras akseleracijos reikšmes ir stiprumą visuose ašyse padaro panašų. Nors ir didžiausia akseleracija vyksta z ašyje, tačiau aukštų dažnių filtras praleidžia tik aukštus dažnius, todėl šios akseleracijos reikšmės yra stipriai

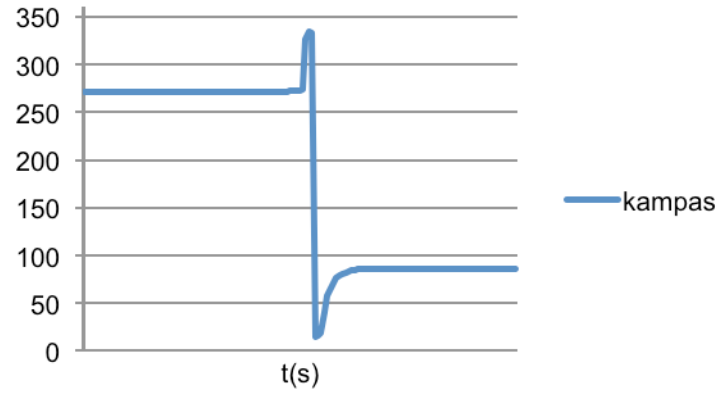
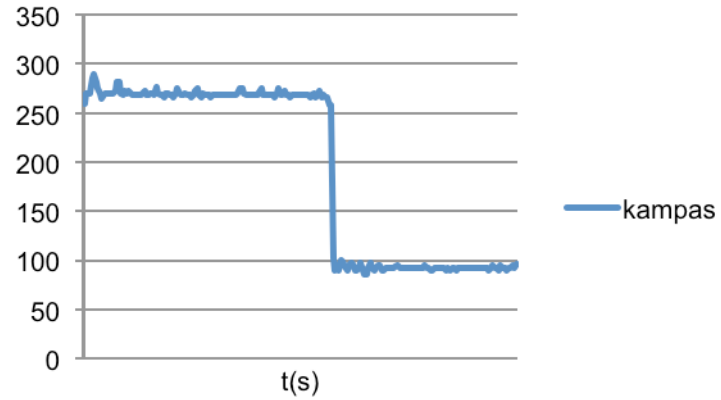
filtruojamos ir visuose ašyse stiprumas yra vienodas. Pagal apibrėžtus reikalavimus, svarbu išlaikyti stiprumo dydį, nukreiptą tam tikra kryptimi, todėl šiuo atveju tinkamesnis yra žemų dažnių filtras.

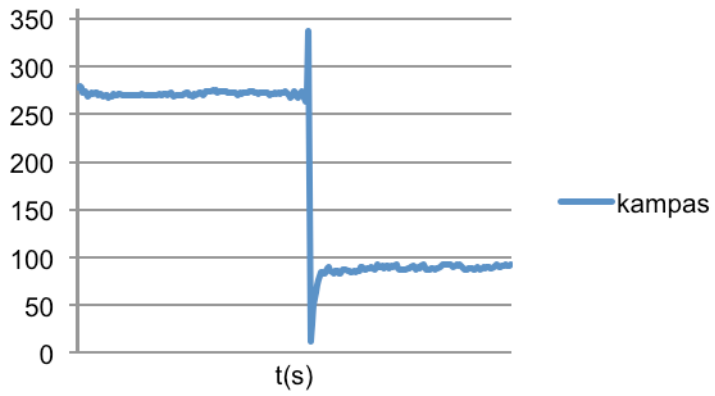
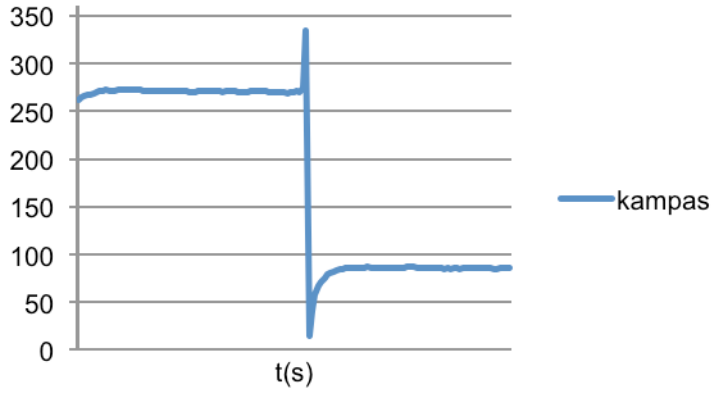
“Gimbal lock” problema

Ši problema detaliau aprašyta 2.3.3 skyriuje. Problema buvo sprendžiama pritaikius paprastą tiesinę interpoliaciją arba interpoliaciją pasitelkiant kvaternionus tarp dviejų kampų.

Tyrimas buvo atliktas žaidimo objekto judesį iš pradžių nukreipiant į viršų, o po to staiga pakeičiant kryptį į apačią. 6.3 lentelė. parodyti galutinio kampo priklausomybė nuo laiko judėjimo būsenoje.

6.3 lentelė. Posūkio kampo nustatymo tyrimas

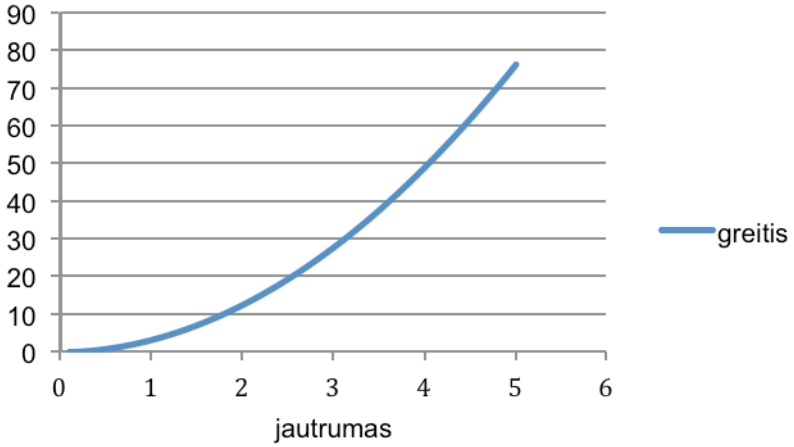
Metodas	Grafikas
Siekiama kampo priklausomybė	
Nenaudojama jokia interpoliacija	

Naudojama tiesinė interpoliacija	
Naudojama interpoliacija pasitelkiant kvaternionus	

Iš pastarųjų lentelių matome, kad nenaudojant jokios interpoliacijos tarp kampų, pastebimas valdomo objekto stiprus drebėjimas. Staiga pakeitę objekto judėjimo kryptį, iš karto pakeičiame ir kampą. Toks judėjimas yra nenatūralus ir netinkamas žaidimo valdymui. Pritaikę tiesinę interpoliaciją tarp dviejų kampų gavome natūralesnį judėjimą, tačiau drebėjimas vis dar egzistuoja. Pritaikę interpoliaciją pasitelkiant kvaternionus, gavome arti siekiamos kampo priklausomybės. Drebėjimas buvo beveik panaikintas, o staiga pakeista judėjimo kryptis bendrą kampą leido pakeisti palaipsniui. Kuriant akselerometru paremtą valdymą, svarbu judėjimą padaryti sklandų ir natūralų. Akivaizdu, kad čia geriausiai tinka interpoliacija pasitelkiant kvaternionus.

Akselerometro jautrumo įtaka valdymui

Valdymo patogumas labai priklauso nuo akselerometro jautrumo. Kuo jautrumas didesnis, tuo didesnis valdomo objekto judėjimo greitis. Atlikus bandymus, nustatyta, kad jautrumo koeficientai turėtų būti (0,5] intervale. Šiame intervale ir patikrinta greičio priklausomybė nuo jautrumo stiprumo. Pasirinkus didesnį jautrumą, valdymas tampa nebekontroliuojamas.

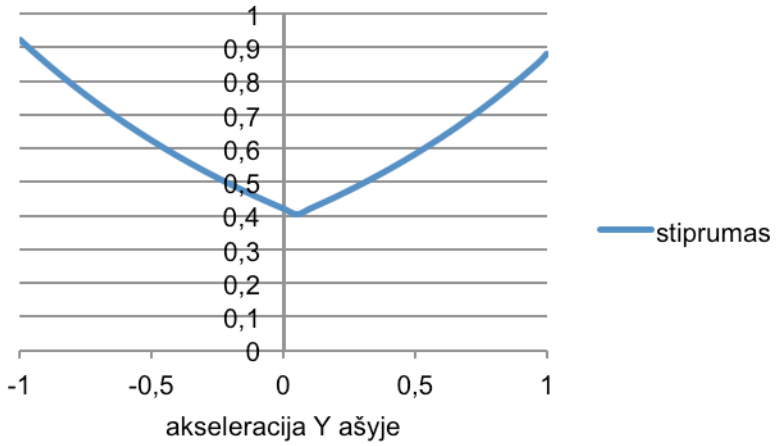


6.1 pav. Greičio priklausomybė nuo jautrumo

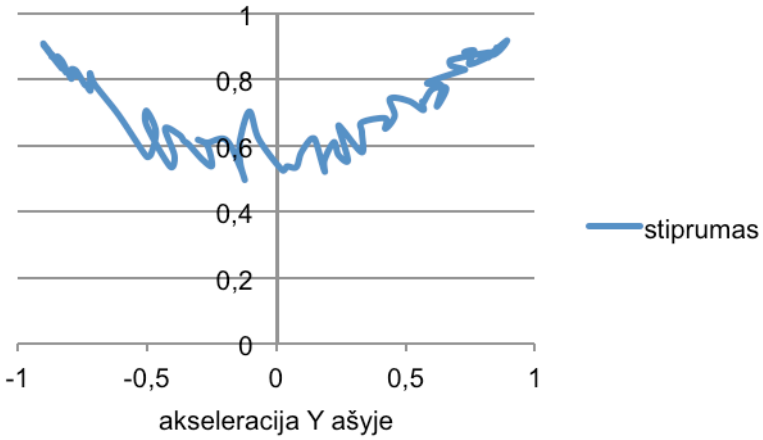
Iš 6.1 pav. matome, kad greitis funkciškai priklauso nuo jautrumo, o kuo didesnis jautrumas, tuo valdomas objektas įgyja didesnį pagreitį ir juda greičiau.

Judėjimo stiprumo priklausomybė nuo įrenginio pakreipimo

Objekto judėjimas priklauso nuo įrenginio pakreipimo stiprumo. Idealiu atveju judėjimo stiprumo vektoriaus dydis pavaizduotas 6.2 pav.



6.2 pav. Judėjimo stiprumo priklausomybė nuo įrenginio pakreipimo idealiu atveju



6.3 pav. Judėjimo stiprumo priklausomybė nuo įrenginio pakreipimo

Dėl akselerometro triukšmo, tam tikrais momentais atsiranda stiprumo trukdžių (6.3 pav.). Matome, kad kuo akseleracija didesnė (-1 ir +1 yra didžiausia akseleracija įrenginį palenkus į priekį arba atgal), tuo didesnis judančio objekto stiprumo vektorius. Bendram judėjimo greičiui triukšmas, pavaizduotas grafike, didelės įtakos neturi, nes stiprumo įtaka judėjimo greičiui yra funkcinė.

Žaidimo valdymo nustatymo problemos

Akselerometro valdymo teisingam veikimui reikalingas tam tikrų parametrų pradinių reikšmių nustatymas – kalibravimas. Jis priklauso nuo to, kokioje pozicijoje yra laikomas įrenginys. Dažniausiai kiekvienam individualiam žmogui, kad galėtų patogiai valdyti žaidimo objektą, šie parametrai yra skirtingi.

Pagrindinė akselerometru valdomo žaidimo valdymo nustatymo problema – kaip pateikti valdomo objekto kalibravimą vartotojui, kad jis iš karto suprastų ką ir kaip reikia daryti. Pradinė mintis buvo laikytis minimalaus žingsnių skaičiaus žaidimo pradžiai pasiekti ir tik jį startavus, prietaiso kalibravimą nustatyti automatiškai atsižvelgiant į tuo metu esančią laikymo poziciją. Toks sprendimas yra geras, kai žinai, kaip viskas veikia. Atlikus žaidimo bandymus su keliais vartotojais išryškėjo tokio sprendimo problemos:

- Tik pradėjus žaidimą valdymas tiesiog veikia, tačiau pakeitus prietaiso poziciją, pvz. iš standartinės prieš veidą padėjus išilgai ant stalo, vartotojai tiesiog nebežino ką daryti, nes valdomas objektas pradeda elgtis keistai ir nevaldomai.
- Labiau patyrę žaidėjai tokiu atveju ieško kalibravimo nustatymų, tačiau šio sprendimo atveju, mes kalibravimą atliekame automatiškai, todėl vėl neaišku, kada jis atliekamas.
- Žvelgiant labiau iš socialinės aplinkos pusės akivaizdu, kad žaidimas būtų valdomas bet kaip, o bet koks prietaiso pozicijos pakeitimas tiktai sukeltų valdymo sumaištį.

Kas būtų, jei apie automatinį kalibravimą praneštume vartotojui? Tokio pokyčio rezultatai:

- Patyrę vartotojai, suprantantys techninį kalibravimo poreikį iš karto suprato kaip viskas veikia.
- Kiti vartotojai tiesiog nesuprato, ką reiškia valdymo kalibravimas.

Iš bandymų rezultatų galima daryti tokias išvadas:

- Be tinkamo paaiškinimo automatinis kalibravimas gali vartotojus priversti manyti, kad valdymas tiesiog nebeveikia.
- Vartotojas pats nori nustatyti, kaip jis valdys žaidimo objektą, todėl automatinis nustatymas čia netinka.
- Akselerometru paremtas valdymas dažnai reikalauja gana tikslaus valdymo ir paliekant automatinį kalibravimo nustatymą tokį tikslumą išgauti labai sunku.

Tokių problemų sprendimas, kuris ir įgyvendintas, yra toks

1. Žaidimo pradžioje pateikti fiksuotą kalibravimo padėtį.
2. Žaidimo metu vienu mygtuko paspaudimu nustatyti naują kalibravimo padėtį.

6.2 Kokybės kriterijų įvertinimas

2.11 skyriuje buvo apibrėžti rezultato kokybės kriterijai ir suskirstyti į tokias kategorijas: veikimas, pritaikymas, paprastumas, patikimumas, plečiamumas ir tęstinumas. Toliau pateikiamas šių kokybės kriterijų įvertinimas sukurtam galutiniam metodui. Kokybės kriterijaus pilną kokybės atitikimą žemėsime “+” ženklu, dalinį kokybės atitikimą žymėsime “+-“ ženklu, o netenkinantį kokybės žymėsime “-“ ženklu.

6.4 lentelė. Veikimo kriterijų įvertinimas

Kriterijus	Įvertinimas	Pastabos
Akselerometro duomenų triukšmo panaikinimas	+	Žr. 6.1 lentelė., 6.2 lentelė.
Teisingas posūkio kampo nustatymas	+	Žr. 6.3 lentelė.
Teisingas judėjimo krypties nustatymas	+	Žr. 6.3 lentelė.
Sklandus galutinis objekto judėjimas ekrane	+	Žr. 6.1 lentelė., 6.2 lentelė., 6.3 lentelė.

6.5 lentelė. Pritaikymo kriterijų įvertinimas

Kriterijus	Įvertinimas	Pastabos
Objektą galima judinti visomis kryptimis	+	Patvirtinta bandymų metu
Objektą galima valdyti imituojant vairo sukimą	+	Patvirtinta bandymų metu
Objektą galima valdyti gestų pagalba	+-	Galimi tik paprasti gestai, pvz. staigus įrenginio judinimas aukštyn-žemyn

6.6 lentelė. Patikimumo kriterijų įvertinimas

Kriterijus	Įvertinimas	Pastabos
Objektas visada juda ten, kur tikimasi	+	Patvirtinta bandymų metu
Pritaikius akselerometro duomenų filtravimą, visiškai panaikinamas triukšmas	+-	Žr. 6.1 lentelė., 6.2 lentelė., 6.3 lentelė. Minimalus triukšmas išlieka
Metodas naudoja efektyvų atminties valdymą	-	Atminties valdymui skirta mažiau dėmesio, ilgiau veikiant sistemai, pastebėtas jos sulėtėjimas

6.7 lentelė. Plečiamumo, tęstinumo kriterijų įvertinimas

Kriterijus	Įvertinimas	Pastabos
Teoriškai aprašytas metodas pritaikomas bet kokiai sistemai	+	Žr. 4 skyrių
Sukurtas valdymo metodas paveldimas ir tokiu būdu praplečiamas	+-	Galima paveldėti, tačiau tinka tik konkretiems atvejams, kada naudojamas “cpShapeNode” fizikinis kūnas. Žr. 4.3 skyrių
Sukurtas valdymo metodas neriboja valdymo krypties	+	Patvirtinta bandymų metu
Yra galimybė keisti jautrumo, filtravimo kalibravimo parametrus	+	Realizuota sistemoje
Kodas aiškus ir struktūrizuotas	+	Žr. 5 skyrių

Iš kriterijų įvertinimo galime pastebėti, kad realizuotas galutinis valdymo metodas beveik atitinka užsibrėžtą kokybės lygį. Tam tikri kriterijai kokybę atitinka tik dalinai, tačiau galutiniam vartotojo pasitenkinimui labai didelės įtakos nedaro, nes dažniausiai nėra pastebimi neatlikus išsamaus eksperimentavimo ir duomenų įvertinimo.

6.3 Taikymo rekomendacijos ir tolimesnės plėtojimo galimybės

Darbo metu sukurtas metodas buvo orientuojamas į žaidimų valdymo palengvinimą. Pagrindinė šio metodo savybė – visapusiškas, sklandus žaidimo objekto valdymas su galimybe pritaikyti paprasčiausius gestus. Taigi, šį metodą būtų galima pritaikyti:

- žaidimo objekto valdymui įvairiomis kryptimis,
- žaidimo objekto valdymui, panaudojant paprasčiausius gestus,
- lenktynių simuliatoriuje vairo sukimo imitacijai.

Metodas buvo kuriamas iOS operacinei sistemai, kuri naudojame iPhone mobiliajame įrenginyje, todėl taikomi filtrai, įvairūs koeficientai kitose sistemose gali skirtis ir reikalauja papildomo tyrinėjimo.

Sukurtas metodas šiuo metu geriausiai veikia objekto valdymui įvairiomis kryptimis, tačiau gali atpažinti tik pačius paprasčiausius gestus. Siekiant toliau plėtoti metodą, būtų galima įgyvendinti ir įvairių gestų atpažinimą, pvz. apskritimo, kvadrato, trikampio ir kt. Toks funkcionalumas padėtų metodą pritaikyti daug įvairesniuose žaidimuose.

6.4 Eksperimento rezultatai

Eksperimento metu buvo patikrintas sukurto metodo korektiškas veikimas, įvertinta kokybė užsibrėžtų kriterijų kontekste. Metodui sukurti buvo naudojami įvairūs algoritmai, ieškoma įvairių koeficientų, sprendžiamos objekto pasukimo kampo problemos. Eksperimentas padėjo pasirinkti geriausius būdus šioms problemoms spręsti įvertinus ir palyginus idealius ir gautuosius skaičiavimus. Eksperimento metu buvo:

- pasirinktas žemų dažnių filtras akselerometro triukšmingiems duomenims filtruoti,
- pasirinktas posūkio kampo interpoliacijos būdas panaudojant kvaternionus,
- patikrinta akselerometro jautrumo įtaka valdymui,
- nustatyta judėjimo stiprumo priklausomybė nuo įrenginio pakreipimo kampo.

7 Išvados

1. Dėl unikalių akselerometro savybių ir savo mažo dydžio jis dažniausiai naudojamas įvairiuose mobiliuosiuose įrenginiuose, o čia gali būti pritaikytas supaprastintam žaidimų valdymui.
2. Akselerometras paprastas funkcijas gali praplėsti ar papildyti naujomis, taip suteikdamas vartotojui daugiau laisvės, o tai leidžia kasdienes užduotis atlikti sparčiau ir paprasčiau.
3. Egzistuoja dvi svarbios akselerometro panaudojimo problemos: triukšmingi akselerometro duomenys ir pasukimo kampo interpoliacijos sunkumai.
4. Vienintelė iPhone programų kūrimui naudojama programavimo aplinka yra Xcode, o programuojama Objective-C programavimo kalba naudojant iOS SDK biblioteką. Šios bibliotekos pagrindinė klasė, skirta akselerometro valdymui, yra UIAccelerometer.
5. Sukurtas detalus projektas, funkciniai ir nefunkciniai reikalavimai padeda iš anksto turėti bendrą sistemos vaizdą, kuris leidžia sistemą kurti greičiau ir užtikrina, kad nebūtų nukrypimų nuo norimos galutinės realizacijos.
6. Sukurtas naujas metodas akselerometro duomenis pritaikyti žaidimo objekto valdymui įvairiomis kryptimis, panaudojant paprasčiausius gestus, imituojant vairo pasukimą. Metodas panaikina visas triukšmo ir kampų interpoliacijos problemas. Jo pagrindu sukurtas žaidimo prototipas.
7. Eksperimentu patikrintas sukurto metodo korektiškas veikimas, pasirinkti geriausi algoritmai ir koeficientai, leidžiantys praktiškai gautus duomenis priartinti prie teorinių duomenų, teigiamai įvertinti užsibrėžti kokybės kriterijai.

8 Literatūra

1. Jill Attewell. Mobile technologies and learning. Technology Enhanced Learning Research Centre, 2004.
2. Tamas Vajk, Paul Coulton, Will Bamford, Reuben Edwards. Using a Mobile Phone as a "Wii-like" Controller for Playing Games on a Large Public Display. International Journal of Computer Games Technology, 2008.
3. Įvairūs akselerometro panaudojimo būdai. [Žiūrėta 2009-10-10] Prieiga per internetą: http://www.creativeapplications.net/iphone/10-creative-ways-to-use-the-accelerometer-iphone/
4. Steffen Itterheim. Learn iPhone and iPad cocos2d Game Development. Apress 2010.
5. Mark Lemkin, Bernhard E. Boser. A Three-Axis Micromachined Accelerometer with a CMOS Position-Sense Interface and Digital Offset-Trim Electronics.
6. Carlos Vallin. MEMS accelerometers as motion tracking devices. Section L01 CP I/O Group.
7. Texas Instruments akselerometro veikimo gidas. [Žiūrėta 2009-10-11] Prieiga per internetą: http://www2.usfirst.org/2005comp/Manuals/Acceler1.pdf
8. Piezoelectric sensors. [Žiūrėta 2009-10-20] Prieiga per internetą: http://www.piezocryst.com/piezoelectric_sensors.php
9. How to design an accelerometer. [Žiūrėta 2009-11-10] Prieiga per internetą: http://www.memsuniverse.com/?page_id=1548
10. iPhone 3GS dissasembly guide. [Žiūrėta 2010-01-08] Prieiga per internetą: http://www.rapidrepair.com/guides/iphone-3g-s-repair/iphone-3g-s-dissasembly-repair-guide.html
11. iPhone reference library. Prieiga per internetą: http://developer.apple.com/iphone/library/navigation/index.html
12. Accelerometer testing. [Žiūrėta 2010-04-29] Prieiga per internetą: http://groklab.org/handhygiene/2010/02/04/accelerometer-testing/
13. Dave Mark, Jeff LaMarche. Beginning iPhone development: Exploring the iPhone SDK. Apress, USA, 2009.
14. Introduction to Objective-C Programming language: Prieiga per internetą: http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

15. AccelerometerGraph sample code. Apple, 2010. [Žiūrėta 2011-01-10] Prieiga per internetą: http://developer.apple.com/library/ios/samplecode/AccelerometerGraph/Introduction/Intro.html
16. Marco Klingmann. Accelerometer-Based Gesture Recognition with the iPhone. Goldsmiths Univerity of London, 2009.
17. Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, Michael L. Littman. Activity Recognition from Accelerometer Data. Department of Computer Science Rutgers University, Piscataway, 2005.
18. M. J. Mathie, J. Basilakis, B. G. Celler. A system for monitoring posture and physical activity using accelerometers. Biomedical Systems Laboratory, University of New South Wales, Sydney, Australia.
19. Thongchai Kolyutsakul, Nattapong Kurupraditwong, Natasha Dejdumrong. An Educational iPhone Game:Fishing Adventure using iPhone SDK and Cocos2D Library. Department of Computer Engineering, 2010.

9 Terminų ir santrumpų žodynas

iOS – Apple kuriama ir palaiko operacinė sistema mobiliems įrenginiams.

iPhone – Apple kuriamas mobilus įrenginys, palaikantis iOS operacinę sistemą.

Wi-Fi – bevielio ryšio technologija.

Interpoliacija – Tai toks atskirų taškų apjungimo procesas, kad gautume tinkamus duomenų įverčius tarp duotų taškų.

Kvaternionas – hiperkompleksinis skaičius a + bi + cj + dk, kai a, b, c, d — realieji skaičiai, o simboliai i, j, k tenkina tam tikras sąlygas.

Xcode – programavimo aplinka Mac OS X operacinėje sistemoje.

Mac OS X – Apple kuriama ir palaikoma operacinė sistema.

MVC – (angl. Model-View-Controller) programavimo modelis, kuris kodą padalina į tris dalis: modelį, vaizdą ir valdiklį.

Objective-C – Apple kuriama ir palaikoma programavimo kalba.

C++ – Programavimo kalba.

OpenGL – kompiuterinės grafikos specifikacija, palaikoma įvairiuose operacinėse sistemose ir programuojama bet kokia programavimo kalba.

SDK – (angl. Software Development Kit) programavimo bibliotekos rinkinys su apibrėžtomis pagalbinėmis klasėmis.

delegatas – specialus protokolas, leidžiantis naudotis delegato metodais, pvz. akselerometro duomenų atnaujinimo metodu.

XML – (angl. Extensible Markup Language) W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

MySQL – reliacinių duomenų bazių sistema.

Debesų kompiuterija – (angl. cloud computing) įvairių paslaugų sistema nutolusiame kompiuteryje, kurioms pasiekti reikalingas tik interneto ryšys.

AppStore – Apple programų parduotuvė ir saugykla.

10 Priedai

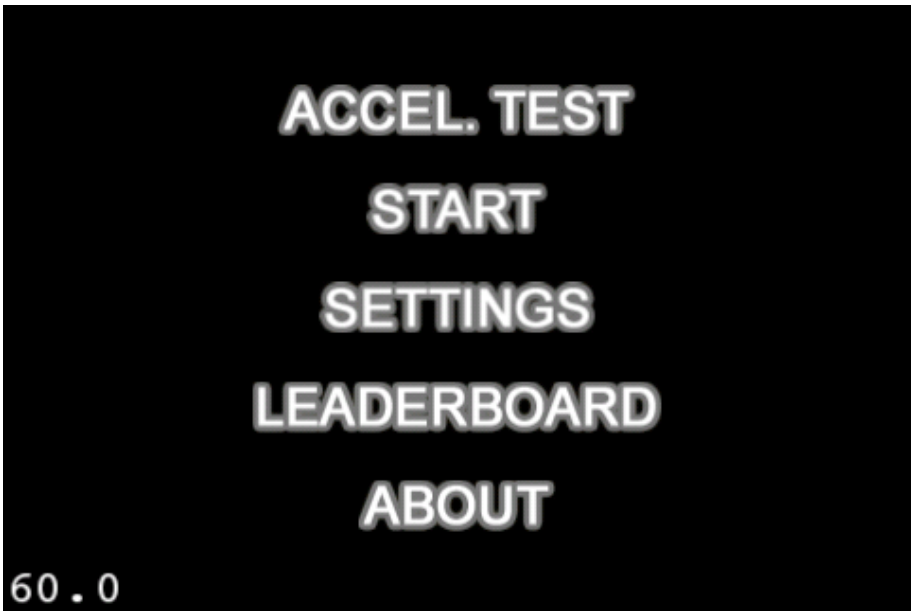
10.1 Vartotojo vadovas

Žaidimas paleidžiamas paspaudus ant žaidimo ikonos pavadinimu “LogBalance”. Iš karto po paspaudimo pasirodo krovimosi langas, kuris reiškia, kad kraunami įvairūs sistemos komponentai ir bibliotekos. (10.1 pav.)

10.1 pav. Sistemos paleidimas ir krovimosi langas

Po žaidimo užkrovimo pasirodo pradinis langas, pavaizduotas 10.2 pav. Iš pradinio lango, paspaudus ant atitinkamo meniu punkto, galima pasiekti šiuos sistemos langus:

- Testavimo aplinka,
- Žaidimas,
- Nustatymai,
- Pasiekimai,
- Apie.



10.2 pav. Pradinis sistemos langas

Pirmą kartą įjungus žaidimą, paprašoma prisijungti prie Apple žaidimų paskyros (10.3 pav.). Čia galima naudoti jau egzistuojančia paskyrą, arba susikurti naują.

10.3 pav. Prisijungimas prie Apple paskyros

Jei vartotojas pasirenka naudoti jau egzistuojančią paskyrą, paprašoma įvesti tokius šios paskyros prisijungimo duomenis (10.4 pav.):

- elektroninio pašto adresą,
- slaptažodį.

10.4 pav. Egzistuojančios paskyros naudojimas

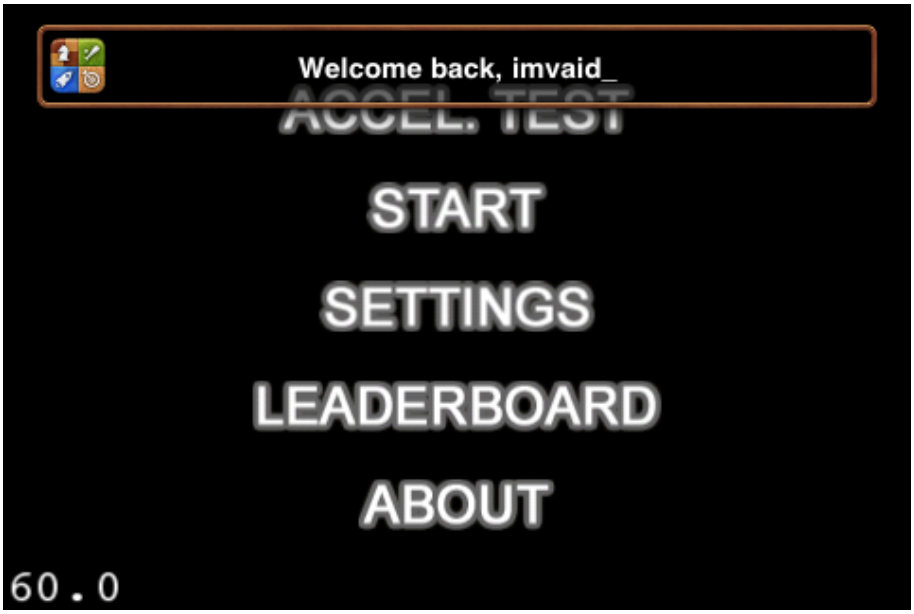
Jei vartotojas pasirenka sukurti naują paskyrą, atidaromas vartotojo kūrimo vedlys (10.5 pav.).

10.5 pav. Naujos paskyros kūrimo etapai

Paskyros kūrimas susideda iš tokių žingsnių:

- šalies pasirinkimas,
- gimimo dienos nustatymas,
- pagrindinių duomenų įvedimas: vardas, pavardė, el. paštas, slaptažodis.
- taisyklių patvirtinimas.

Po sėkmingo prisijungimo parodomas pranešimas (10.6 pav.).



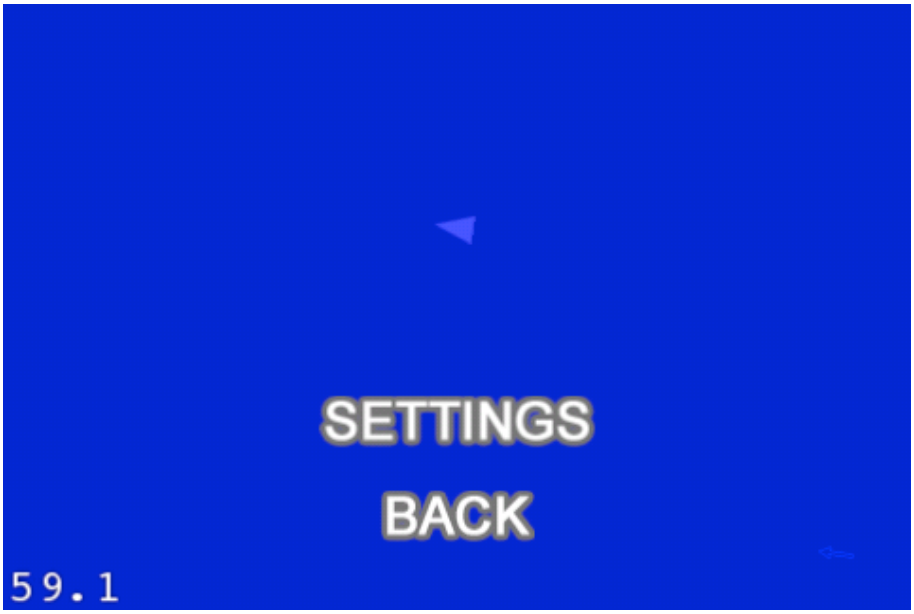
10.6 pav. Pranešimas apie sėkmingą prisijungimą

Vartotojui atidarius testavimo langą matomi tokie pagrindiniai komponentai (10.7 pav.):

- valdomas objektas,
- stiprumo ir krypties vektorius.

10.7 pav. Testavimo aplinka

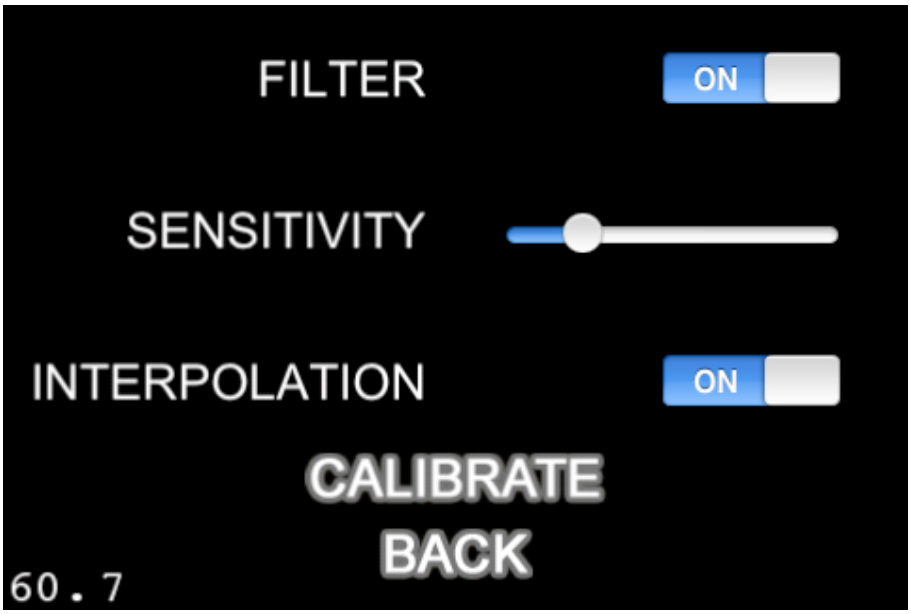
Paspaudus bet kur testavimo lange, testavimo aplinka sustabdoma. Čia galima iškviesti nustatymų langą arba vėl sugrįžti į testavimo aplinką. (10.8 pav.)



10.8 pav. Testavimo aplinkos sustabdymo langas

Atidarius nustatymų langą, galima koreguoti tokius nustatymus (10.9 pav.):

- įjungti/išjungti duomenų filtravimą,
- nustatyti akselerometro judėjimo jautrumą,
- įjungti/išjungti posūkio kampo interpoliaciją,
- rankiniu būdu atlikti akselerometro kalibravimą.

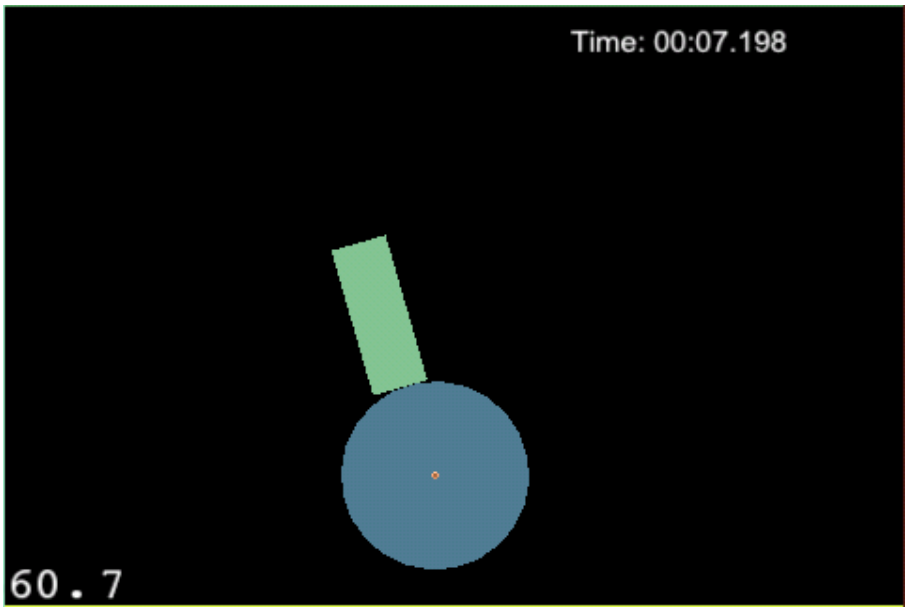


10.9 pav. Nustatymų langas

Vartotojui atidarius žaidimo langą matomi tokie pagrindiniai komponentai (10.10 pav.):

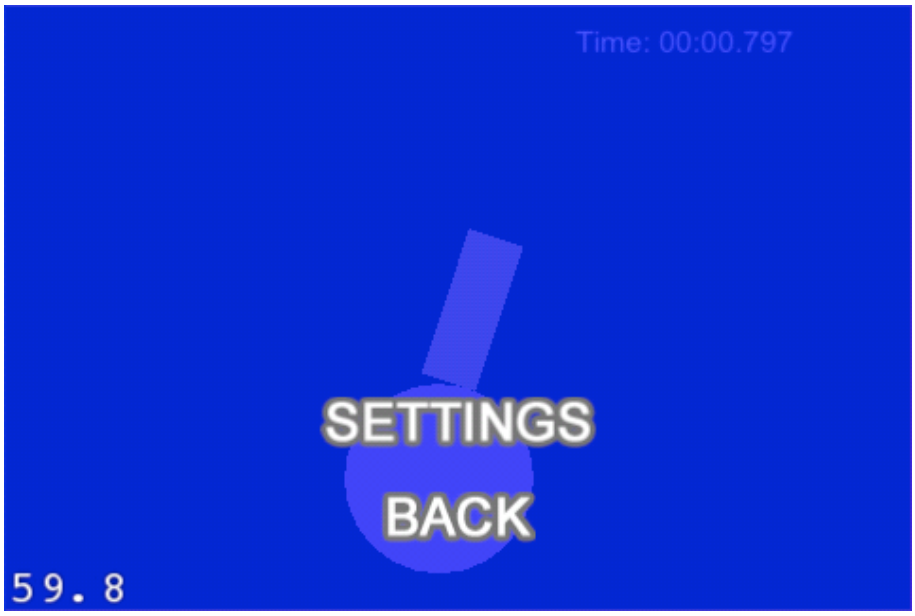
- akselerometru valdomas žaidimo objektas (stačiakampis), turintis fizinę masę,
- besisukantis ant ašies susidūrimo objektas (apskritimas), turintis fizinę masę,
- žaidimo laikas, matuojantis balansavimo trukmę.

Vartotojo tikslas – kuo ilgiau balansuoti žaidimo objektą (stačiakampį) ant besisukančio apskritimo.



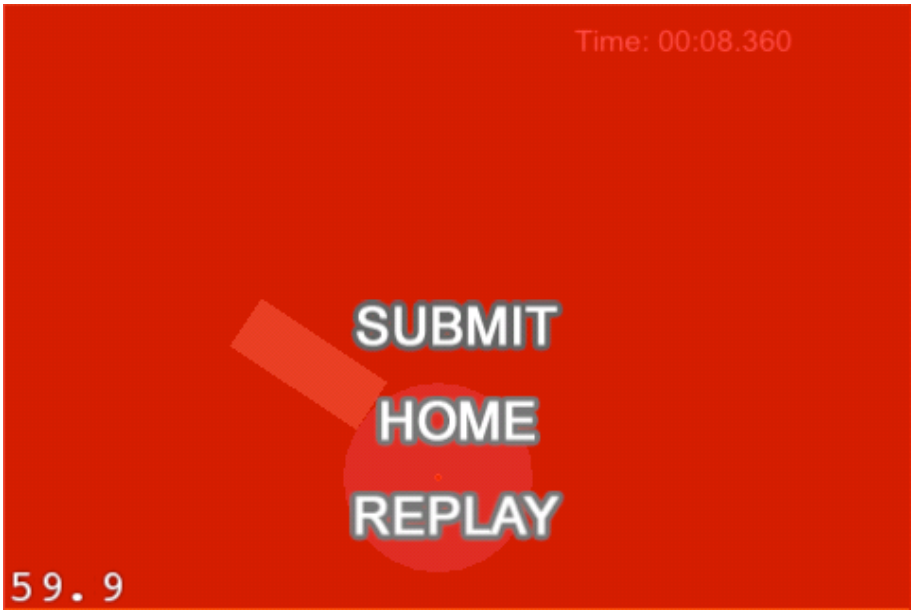
10.10 pav. Pradinis žaidimo langas

Paspaudus bet kur žaidimo lange, žaidimas, o tuo pačiu ir bėgantis laikas sustabdomi. Čia galima iškviesti nustatymų langą arba vėl sugrįžti į žaidimo aplinką.



10.11 pav. Žaidimo sustabdymo langas

Kada žaidimo objektas kritimo metu pasiekia tam tikrą ribą, kai atstumas tarp jo ir besisukančio apskritimo yra didesnis už bandymų metu nustatytą koeficientą, iškviečiamas žaidimo pabaigos langas (10.12 pav.). Tokiu atveju fiksuojamas galutinis žaidimo laikas, kurį galima publikuoti į rekordų duomenų bazę. Iš žaidimo pabaigos lango taip pat galima iš naujo pradėti žaidimą arba grįžti į pradinį langą.



10.12 pav. Žaidimo pabaigos langas

Atidarius laiko publikavimo langą, rodomi geriausi pastarosios dienos, savaitės ir mėnesio rezultatai (10.13 pav.). Jeigu prisijungusio žaidėjo pasiektas laikas nėra geresnis už jo geriausią laiką, šis rezultatas neskelbiamas. Iš šio lango galima grįžti į prieš tai buvusį aktyvų langą.

10.13 pav. Geriausių rezultatų langas

Pasirinkus apie langą, rodoma pagrindinė informacija apie sistemos kūrėją (10.14 pav.). Iš apie lango galima grįžti atgal į pradinį langą.

10.14 pav. Apie langas