VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

PRAKTIKOS ATASKAITA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Praktiką atliko: | **Sigitas Ambrozaitis +37067958761** | | | | | | |
| (studento vardas, pavardė) | | | | | (tel nr.) (parašas) | | |
|  | | | | |  | | |
|  | **Informatika, 4 kursas** | | | | | | |
|  | | (studijų programa, pakopa, kursas) | | | | | |
|  | | | | | |  | |
|  | | | | | |  | |
| Praktikos institucija: | | | UAB „Nordcurrent Group“ | | | | |
|  | | | | | | (organizacijos pavadinimas) | |
|  | | | | | | | |
| Organizacijos praktikos vadovas: | | | | Projektų ir technologijų direktorius Michail Trofimov | | | |
|  | | | | | | (pareigos, vardas, pavardė) | |
|  | | | | | |  | |
| Organizacijos praktikos vadovo įvertinimas: | | | | |  | | |
|  | | | | | | (įvertinimas, parašas) | |
|  | | | | | |  | |
| Universiteto praktikos vadovas: | | | | Lekt. Irmantas Radavičius | | | |
|  | | | | | | (mokslo laipsnis, vardas, pavardė) | |
|  | | | | | |  | |
|  | | | | | |  | |
|  | | | | | | (parašas) | |
|  | | | | | | | |
|  | | | | Ataskaitos įteikimo data | | |  |
|  | | | | Registracijos Nr. | |  | |
|  | | | | Įvertinimas | |  | |
|  | | | |  | | (data, įvertinimas, parašas) | |

Vilnius, 2020

**Turinys**

[Praktikos vietos aprašymas 3](#_Toc38477076)

[Organizacijos pristatymas 3](#_Toc38477077)

[Darbo sąlygos 3](#_Toc38477078)

[Įvadas 4](#_Toc38477079)

[Teorinis-metodinis skyrius 5](#_Toc38477080)

[Naudota programinė įranga 5](#_Toc38477081)

[Git ir Jira 5](#_Toc38477082)

[Unity ir C# 6](#_Toc38477083)

[Tiriamasis-analitinis skyrius 7](#_Toc38477084)

[Komponento realizacijos iteracija 7](#_Toc38477085)

[Nepriklausomo komponento problema 7](#_Toc38477086)

[Baziniai komponentai 8](#_Toc38477087)

[Mygtukų komponentai 8](#_Toc38477088)

[Slankiklių komponentai 9](#_Toc38477089)

[Slankiklius pildantys komponentai 9](#_Toc38477090)

[Išvados 11](#_Toc38477091)

[Pagrindinės išvados 11](#_Toc38477092)

[Darbo privalumai 11](#_Toc38477093)

[Darbo trūkumai 11](#_Toc38477094)

[Įgytos žinios 11](#_Toc38477095)

# Praktikos vietos aprašymas

## Organizacijos pristatymas

„Nordcurrent“ – tarptautinė, bei viena iš didžiausių Lietuvos žaidimų kūrimo ir leidimo kompanijų. Savo veiklą pradėjo nuo 2002 metų pavadinimu „Ivolgamus“, kaip konsolinių žaidimų kūrimo bendrovė. Vėliau 2007 metais bendrovė įkūrė žaidimų kūrybos ir leidybos padalinį „Nordcurrent“. O nuo 2010 specializuojasi „freemium“ (iš dalies nemokamų produktų ir paslaugų tiekimo verslo modelis) ir „casual“ (orientuotas į masinę auditoriją) multiplatforminių žaidimų kūrime.

Populiariausi Nordcurrent žaidimai yra kulinarinis žaidimas „Cooking Fever“, kuris jau turi beveik 250 milijonų atsisiuntimų, paslėptų objektų žaidimas „Murder In the Alps“, bei „multiplayer“ šaudyklė „Sniper Arena“.

## Darbo sąlygos

„Nordcurrent“ yra įsikūręs dviejuose pastatuose, Naujamiestyje esančiame verslo komplekse „Verslo trikampis“. Darbas vyksta nuo 9h iki 18h, pietų pertrauka būna tarp 13h ir 14h. Ne darbo metu įmonė suteikia galimybę pažaisti įvairius stalo žaidimus, stalo tenisą, stalo futbolą. Taip pat galima išbandyti tiek senus konsolinius žaidimus, tiek naujausios kartos VR žaidimus. Įrengta knygų biblioteka, apie kompiuterinių žaidimų kūrimą, suteikia galimybę toliau gilinti savo žinias.

Darbuotojui standartiškai suteikiama asmeninė darbo vieta, užtikrinanti patogų ir efektyvų darbą. Standartiškai yra suteikiama spartūs kompiuteriai, su aukštos raiškos ekranais, patogios kėdes, bei spintelės asmeniniams daiktams, prieiga prie versijavimo, bei projekto valdymo sistemų. O priklausomai nuo darbo pobūdžio yra suteikiama papildoma techninė, bei programinė įranga.

# Įvadas

**Praktikos tema: ž**aidimų programavimas C# kalba.

**Uždaviniai:**

1. Žaidimų algoritmų pritaikymas
2. C# kalbos įgūdžių tobulinimas

Žaidimų kūrime labai svarbus yra komandinis darbas, nustatytų tikslų pasiekimas numatytu laiku, problemų indentifikaciją, jų aptarimas ir greitas sprendimo radimas. Taip pat žaidimuose yra labai svarbu rašyti efektyvų kodą, dėl to įmonė praktikos metu iškėlė šiuos tikslus:

* Įsisavinti darbo eigos principus pagal „Agile“ programų kūrimo metodologiją.
* Išmokti naudotis versijų kontrolės sistema „Git“.
* Išmokti naudotis projektų valdymo sitema „JIRA“
* Praplėsti žinias, susijusias su žaidimų algoritmais.
* Išmokti rašyti efektyvų ir resursus taupantį C# kodą.

Praktikos gauta užduotis – realizuoti „Unity“ žaidimų variklyje naudojamus vartotojo sąsajos komponentus. Šie komponentai yra dviejų tipų, savarankiški, ir papildomieji. Savarankiški komponentai turėjo veikti nepriklausomai nuo jokių kitų komponentų, tuo tarpu papildomieji, priklausomi nuo savarankiškųjų, tačiau tik pildo jų funkcionalumą. Pagal užduotį reikėjo realizuoti įvairius mygtukus ir slankiklius. Taip pat papildomų komponentų, kurie turi išplėsti nurodytų pagrindinių komponentų funkcialumą.

1. Mygtukai
   1. Vieno paspaudimo
   2. Dviejų paspaudimu
   3. Pasirinkimo
2. Slankikliai
   1. Paprastas
   2. Lenktas
   3. Progreso juosta

# Teorinis-metodinis skyrius

## Naudota programinė įranga

Praktikos metu teko naudotis šia programine įranga:

* „Unity“
* „Git“
* „Jira“
* „Visual Studio 2019“

Šiai praktikai atlikti labiausiai padėjo objektinio programavimo žinios, gautos is universitetinio kurso „Objektinis Programavimas C++“, duomenų struktūrų žinios gautos iš „Duomenų Struktūros ir algoritmai“, sugebėjimas skaičiuoti grafines viršūnes iš „Kompiuterinė Grafika“.

Labai daug patarimų apie efektyvų ir švarų kodą suteikė patyrę projekto programuotojai.

Naudingų benrų žinių apie pačio žaidimo kūrimo procesą buvo gauta iš „Udemy“ kurso „ [Unreal Engine C++ Developer: Learn C++ and Make Video Games](https://www.udemy.com/course/unrealcourse/)“

## Git ir Jira

Kadangi standartiškai kiekvienas IT projektas naudoja kokią nors versijavimo sistemą, ir projektų valdymo sistemą, praktikos metu pirmiausia buvau apmokytas naudotis „Git“ ir „Jira“ sistemomis. „Git“ – versijų kontrolės sistema leidžianti peržiūrėti failų pasikeitimus, juos saugoti, šalinti, perkelti, ar kitaip jais manipuliuoti. Ši sistema palengvina darbą, kai prie to pačio projekto dirba keletas darbuotojų. „Jira“ – projekto valdymo bei klaidų sekimo sistema, leidžianti nesunkiai skirstyti užduotis pagal jų prioritetus, sekti projekto progresą, bei registruoti ir spręsti esamas klaidas.

## Unity ir C#

Paprastai kompanija savo žaidimų kūrimui naudoja vidinį žaidimo variklį, tačiau visai nesenai pradėjo bandyti „Unity“ žaidimų variklį. Projektas prie, kurio buvau priskirtas būtent ir yra vienas iš naujų projektų, kuris naudoja „Unity“. Šio žaidimo variklio pagrindas parašytas naudojant C++ kalbą, tačiau žaidimų kurėjam yra atskleista tik C# kalba rašoma sąsaja.

Vienas iš šio variklio privalumų yra daugelio platformų palaikymas, kas leidžia vieną kartą parašyti kodą ir paleistį jį ant daugelio platformų. Taip pat jis turi savo editorių, kuris ne tik turi visus dažniausiai reikiamus įrankius žaidimo kūrimui (scenos kūrimas, resursų valdymas, žaidimo testavimas, debugginimas...),bet ir galimybę pasirašyti savo įrankius specifinėms žaidimo reikmėms.

Svarbiausia variklio kocepcija, tai „GameObject“. Kiekvienas objektas žaidime yra „GameObject“. Iš esmės „GameObject“ nieko nedaro, tai tik laikikliai komponentams kurie realizuoja tikrą funkcialumą. Komponentai tai atskirios klasių instancijos, būtent jie ir nurodo, kaip „GameObject“ elgiasi. „Unity“ variklis siūlo didelę dalį standartinių komponentų ir galimybę juos kurti patiems.

Taip pat svarbi koncepcija yra „ScriptableObject“ tai pavieniai komponentai, kurie nėra priskiriami „GameObject“, tačiau jų instancijos yra kuriamos, kaip papildomi žaidimo failai .asset formatu. Šios instancijos yra skaitomos kaip globalūs kintamieji, o adresus į juos galima nurodyti Editoriuje, arba užkrauti pagal failo kelią failų sistemoje.

# Tiriamasis-analitinis skyrius

## Komponento realizacijos iteracija

Kadangi reikiamų realizuoti komponentų buvo daug, visų komponentų realizacijai buvo naudojama ta pati strategija.

1. Iššsiaiškinti komponento funkcialumą. Tai aptarinėjau kartu su žaidimų dizaineriu. Šio proceso metu buvo pilnai išsiaiškinamas komponento funkcialumas, kaip jis bus naudojamas ir kokios jo limitacijos. Paruošiama pradinė komponento koncepcija, ir galutinio varianto vizija.
2. Komponento realizacijos planavimas, sudaroma UML diagrama, su pagrindinėmis klasėmis, jų metodais bei kintamaisiais. Taip pat suplanuojama, kurie skaičiavimai bus atliekami runtime, kurie edit-time.
3. Komponento realizacija, pagal sutartą funkcionalumą ir UML diagramą.
4. Kodo peržiūra ir klaidų taisymas.

## Nepriklausomo komponento problema

Planuojami komponentai turėjo veikti nepriklausomai nuo kitų žaidimo elemento, tai reiškia, kad pridėjus komponentą, jis pradės savaime veikti, tačiau ir išėmus esamą, žaidimas dėl to veiks toliau, tik be panaikinto komponento funkcialumo. Kitaip tariant visi šie komponentai turėjo veikti kaip nepriklausomi moduliai.

Pasiulytas sprendimo būdas buvo paremtas „Unity“ ScriptableObject ir C# delegatų funkcialumu, kurio įdėja pilnai paaiškinama vienoje iš „Unity“ konferencijos pristatymų „Unite Austin 2017 – Game Architecture with Scriptable Objects“. Scriptable Objektai leidžia laikyt kintamuosius projekte, kaip papildomus žaidimo failus, šie kintamieji savyje turi delegatus, kurie iškviečiami, kai pasikeičia objekte esanti reikšmė. Tokius kintamuosius galima priskirti komponentam, kurie savo ruožtu registruojasi prie pateikto delegato, kai yra sukuriami ir išsiregistruoja, kai yra sunaikinami.

Vienintelis tokios sistemos trūkumas, buvo tai, kad reikėjo rašyti daug pasikartojančio kodo, norint sukurti naujo tipo kintamąjį. Dėl to realizavau kodo generatorių, kuris atlieka visą pasikartojančio kodo generavimą. Nors tai nebuvo viena iš mano užduočių, tai buvo darbas, kuris man vėliau sutaupė labai daug laiko.

## Baziniai komponentai

Kadangi gauti komponentai pagrinde priklausė mygtukams arba slankikliam. Pagrindinis jų funkcialumas kartojasi. Dėl to ir mygtukam ir slankikliam buvo realizuotos bazinės jų klasės.

BaseButton – komponentas, kuris gauna ir apdoroja vartotojo piršto bei pelės paspaudimus. Iš „Unity“ suteiktų metodų, ši klasė sugeba atpažinti, ar žaidėjas esamu metu sąveikavo su mygtuku, ignoruoja keletą paspaudimų ant mygtuko vienu metu, sekdamas, tik tą veiksmą, kuris buvo pirmas pradėtas. Taip pat suskaičiuoja kiek pasikeičia veiksmo pozicija ekrane, bei praneša, kada jis buvo paspaustas ir kada atleistas.

BaseSlider – komponentas, kuris turi minimalius ir maximalius reikšmių rėžius, taip pat reikšmę – „Value, kuri yra tarp nustatytų rėžiu bei progresaą – „Progress“, kuris nurodo, kaip arti reikšmė yra maximalios, iškviečia delegatų veiksmus, kai pasikeičia „Value“. Visi parametrai gali būt nustatyti, tačiau jei keičiasi „Value“, perskaičiuojamas „Progress“, jei keičiasi „Progress, perskaičiuojama „Value“. Rekursijos yra išvengiama. Taip pat suteikia galimybę priregistruoti veiksmus, prie „Value“ ir „Progress“ pasikeitimų.

BaseSliderExtension – komponentas kuris edit-time, susieja save, su esamu BaseSlider komponentu, taip sutaupant resursus paleidus žaidimą. Gyvuoja tame pačiame „GameObject“.

## Mygtukų komponentai

Realizavus bazinę mygtukų klasę, visi kiti realizuojami mygtukai paveldėjo iš BaseButton ir visų skiritngų mygtukų funkcialumą buvo labai nesunku realizuoti.

1. Vieno paspaudimo, atpažinus mygtuko atleidimą iškviečia priskirto ScriptableObjekto veiksmą.
2. Dvieju paspaudimu, atpažinus mygtuko atleidimą, pradedą laukti sekančio mygtuko atleidimo, jei gauna mygtuko atleidimą, iškviečia pirskirto ScriptableObject veiksmą.
3. Pasirinkimo
   1. Prisiregistruoja prie ScriptableObjekto, jei gauta nauja reikšmė sutampa su mygtuko ID, objektas vaizduojamas kaip aktyvus.
   2. Atpažinus mygtuko atleidimą, iškviečia ScriptableObjekto veiksmą paduodamas savo ID kaip parametrą.

## Slankiklių komponentai

Realizuota BaseSlider klase atlieka visas reikalingas reikšmių manipuliacijas, dėl to komponentai, kurie paveldi nuo šios klasės turi realizuoti tik kaip jie veikia, pasikeitus reikšmei.

1. CustomSlider – keičia reikšmę pagal rankenos poziciją slankiklyje, gali veikti horizontaliai arba vertikaliai. Nuskaito vartotojo veiksmus, apriboja juos slankiklio objekto režiuose ir pasirinkta ašimi, gaunant veiksmų poziciją pakeičia slankiklio reikšmę ir perstato rankeną į naujai gautą apribotą poziciją.
2. CurvedSlider – veikia identiškai kaip CustomSlider, tik rankena juda ne horizontaliai/vertikaliai, o apskritimu, maximalų pasisukimo kampą, bei pradžios kampą galima koreguoti.
3. ProgressBar – vienintelis slankiklis, kuris negauna vartotojo įvesties. Pagal slankiklio reikšmę reguliuoja kaip atvaizduojamas yra objektas. Yra porą pasirinkimų kaip jis jį gali atvaizduoti, pirmas, naudojant „Unity“ variklio „Image“ komponento pateiktą sąsają, arba keičiant pačio objekto dydį.

## Slankiklius pildantys komponentai

Šie komponentai pildo slankiklio komponentų funkcialumą. Vieni yra pritaikomi visiems slankikliams, kiti konkretiems slankikliams. Visi šie komponentai paveldi iš BaseSliderExtension komponento.

Visiem slankikliams pritaikomi komponentai

1. SliderRange – žaidimo pradžioje nustato slankiklio minimalią ir maximalią reikšmes.
2. Curve – skaičiuoja progresą ne tiesiogiai pagal reikšmę, o pagal rezultatą funkcijoje f(x), kai x yra reikšmė.
3. SliderValue – atvaizduoja slankiklio reikšmę pateiktame „text“ komponente.

Tik ProgressBar slankikliui pritaikomi komponentai

1. ProgressBarAnimator – reikšmių pakeitimus animuoja, ne iš karto nustant naują reikšmę, o ją interpoliuojant per nustatytą laiką.
2. ProgressBarGradient – keičia objekto spalva priklausomai nuo slankiklio progreso.
3. ProgressBarPulse – keičia objekto spalvą nustatytų intervalu, interpoliuodamas tarp originalios ir nustatytos.

# Išvados

## Pagrindinės išvados

* „Git“ istorija yra melas.
* Pilnai išsiaiškinti reikalavimai mažina klaidų riziką.
* Išankstinis projekto planavimas padeda lengviau vykdyti tolimesnį kodo rašymą.
* Paprastesniu ir intuityvesniu kodu galima išvengti galimų problemų ateityje.
* Reikia stengtis vengti brangių operacijų, tokių kaip atminties alokavimas, nereikalingo naudojimo.
* Bendras problemų sprendimas komandoje veda prie efektyvesnio ir paprastesnio kodo.

## Darbo privalumai

* Duodama laisvė pačiam sugalvoti projekto įgyvendinimą.
* Atliekamas darbas nuosekliai komentuojamas, dažnai pasiūlomi efektyvesni sprendimai, išmokoma švaraus kodo rašymo.
* Įmonė paruošia kompetentingus ir profesionalius darbuotojus.
* Tinkamai atlikta praktika užtikrina tolimesnį darbą įmonėje.

## Darbo trūkumai

* Darbo valandos yra nustaytos ir negalima jų keisti.

## Įgytos žinios

* Įgyti pagrindai dirbant projekte su „Git“ sistema.
* Susipažinau su įmonės veikla ir darbo principais.
* Įtvirtinau C# kalbos žinias.
* Išmokau dirbti komandoje.
* Išmokau optimizuoto ir tvarkingo kodo rašymo.