Универзитет у Београду
Електротехнички факултет


# Развој логичке видео-игре коришћењем функционалног програмирања

Мастер рад

Ментор:
Др Живојин Шуштран, доц

Кандидат:
Кристина Бабић 2021/3193

Београд, Август 2023.

# Апстракт

Игра *Bloxorz* представља надградњу домаћег задатка из предмета Функционално програмирање мастер студија на Електротехничком факултету у Београду. Циљ игре јесте да тестира концентрацију играча и побољша његово логичко размишљање и стратегију. Конкретно, играч треба да у што је мање могуће потеза помери блок из његове почетне до крајње позиције на задатом полигону притом водећи рачуна да блок не испадне са терена или се заустави на делу полигона који пропада. Такмичарски дух подстиче се поређењем освојених бодова на крају игре са бодовима 8 тренутно најбољих играча, односно, додавањем бодова у поменуту листу уколико је играч имао бољи резултат. Додатан део апликације чине опције за креирање и играње новокреираног нивоа које корисник може да користи за тренирање, прављење нових полигона и тражење најбољих решења.

У наставку овог документа биће изложен процес развоја игре, укључујући њен концепт, интеракцију са играчем, као и изазове који су настали при коришћењу ScalaFx библиотеке која је коришћена за израду пројекта.

# САДРЖАЈ

# 1. Увод

Развој компјутерских игара је постао непрестано еволуирајући процес који обухвата различите аспекте програмирања, дизајна и интеракције са играчем. Приликом прављења игара, сусрећемо се са многобројним техничким изазовима.

Логичке видео-игре су жанр видео-игара које укључују решавање логичких задатака [1]. Њима се могу тестирати многе способности решавања проблема, укључујући логику, препознавање образаца, решавање секвенци и попуњавање речи. Иако многе акционе и авантуристичке игре захтевају размишљање како би се прибавили недоступни објекти, права логичка игра се фокусира на решавање логичких задатака као примарне активности. Ове игре често укључују облике, боје или симболе, и играч мора да директно или индиректно манипулише њима по специфичном обрасцу [2].

Оне се разликују од обичних насумичних задатака за решавање, јер обично представљају низ међусобно повезаних изазова сличног типа. Ови изазови могу укључивати препознавање образаца, примену логичких правила или разумевање процеса. Правила игара су обично једноставна, играчи их примењују тако што манипулишу елементима унутар игре. Манипулације обично представљају померање компонената у мрежи или другом интерактивном окружењу. Да би напредовали и победили, играчи морају пажљиво размислити и решити постављене загонетке. Сваки нови изазов постаје тежи како игра напредује, иако неки дизајн игара ублажавају замор играча нудећи лакше нивое између захтевнијих.

Овај мастер рад се фокусира на игру *Bloxorz*. Она представља изазовну геометријску слагалицу која захтева сналажљивост, логичко размишљање и стратегију од играча.

У њему је размотрен процес развоја, укључујући концептуализацију и дизајн игре, избор одговарајућих техничких алатки и технологија, као и имплементацију различитих аспеката игре коришћењем функционалног програмског језика *Scala* и библиотеке *ScalaFX*. Такође, ми ћемо представити изазове који су настали приликом развоја као и решења којима су они превазиђени. Објаснићемо и како је коришћење *ScalaFX* библиотеке допринело креирању сцене и упечатљивог визуелног и играчког искуства.

У наредном поглављу дати су преглед игре и њене најбитније карактеристике. Представљени су значај и опис. Укратко је представљен мотив њеног настанка и дат је преглед постојећих решења.

Треће поглавље описује техничке аспекте и захтеве пројекта са програмским језиком и коришћеним алатима. У оквиру овог поглавља дати су увод у функционално програмирање, *Scala* програмски језик и *ScalaFX* библиотеку која је коришћена за израду пројекта. На крају поглавља представљени су технички захтеви за имплементацију 2Д игре а затим и ставке које је требало размотрити приликом имплементације 3Д модела.

У четвртом делу дати су архитектура софтвера, преглед најважнијих класа и њихових релација. Описани су токови игре у случају такмичарког и тренинг нивоа. Дат је кратак опис *tailrec* рекурзије. Главни детаљи имплементације и техничке карактеристике такође су изложени у овом делу.

Пети део представља упутство за употребу. У њему је објашњено на који начин корисник интерагује са игром кроз детаљан преглед сваке од сцена које се појављују током игре. Као последња ставка овог дела, дато је упутство за креирање новог нивоа.

У шестом делу објашњен је мотив за прављење логичке видео игре. Предложен је списак ствари, односно идеја, које би се могле додати у будућности и које би додатно унапредиле игру. На крају, дати су генералан утисак о игри *Bloxorz* и кратак закључак.

# 2. ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

Ово поглавље представља кратко упознавање са игром *Bloxorz*. У наставку су дати опис игре и преглед њених основних карактеристика, затим мотиви за развој и настанак. На крају, приказан је преглед већ постојећих решења.

## 2.1 Зашто је Bloxorz занимљива и важна игра

Игра *Bloxorz* је занимљива и важна из неколико разлога[3]:

- **Забавно играчко искуство**: *Bloxorz* пружа играчима изазовно и забавно искуство. Комбинује мозгалице, логичке задатке и играње улога, што је чини привлачном широком спектру играча.
- **Развија критичко размишљање:** Игра захтева дубоко размишљање, планирање и решавање проблема. Играчи морају пажљиво испланирати сваки корак и предвидети последице својих акција.
- **Учи основе физике**: Игра *Bloxorz* учи основе физике, као што су гравитација, равнотежа и трење. Играчи морају разумети како ће се блок понашати при сваком потезу.
- **Повећава креативност**: Играчима се даје слобода да експериментишу са различитим стратегијама како би прешли нивое. То подстиче креативност и иновативност.
- **Ментални изазов:** Игра *Bloxorz* може бити прилично изазовна, што помаже у развоју менталних вештина као што су концентрација, стрпљење и упорност.
- **Едукативна вредност**: Игра може имати едукативну вредност, посебно за млађе играче. Може их подучити о основама геометрије, математике и логике.
- **Инспирација за развој игара**: За програмере, *Bloxorz* може бити инспирација за развој сличних игара или додавање сличних елемената у њихове игре.

У суштини, *Bloxorz* је занимљива и важна игра јер спаја забаву са учењем, подстиче менталну агилност и логичко размишљање, те омогућава играчима да се окушају у изазову и унапреде своје вештине док уживају у игри.

## 2.2 Кратак опис игре

Ова популарна мозгалица, која се састоји од различитих нивоа, при чему је сваки са јединственим изазовима и тактикама, има за циљ да се померањем квадратног блока преко равни он доведе из почетне позиције у одређени отвор или циљ на крају нивоа. Основне карактеристике ове игре јесу да се она обично посматра из птичије перспективе, при чему се сваки ниво приказује као равна површина са различитим конфигурацијама полигона и препрека. Блок је обично димензија 2x1 и може се ротирати хоризонтално или вертикално. Плоче на којима блок стоји могу бити различитих облика и врста, укључујући обичне плоче, рупе и друге препреке. За управљање блоком, играч користи тастатуру или екран осетљив на додир (у мобилним верзијама) како би померао квадратни блок по равни. Блок се може кретати напред, назад, лево и десно. Основна физика игре укључује равнотежу и гравитацију. Блок ће пасти ако није постављен на стабилан начин на платформи, тако да играч мора разумети како гравитација утиче на кретање блока. Стога он мора пажљиво да испланира сваки потез како би избегао падање блока у провалију или друге опасности. Игра постепено постаје све тежа како играч напредује кроз нивое. Мапе и препреке постају сложеније, а играч мора уз коришћење научених, развијати и нове стратегије за савладавање све тежих изазова. Поене током игре, играчи углавном зарађују на основу броја потеза или брзине којом заврше ниво. Многе верзије ове игре чувају најбоље резултате како би подстакле такмичарски дух и како би играчи могли да се такмиче са најбољима.

Укратко, у наставку је попис главних елемената игре који се срећу у већини верзија ове игре:

- **Блок** (*Block*): Главни елемент игре је блок, који може бити различитих облика и величина. Циљ је поставити блок тако да усправно стане на отвор на мапи.
- **Мапа** (*Map*): Игра се одвија на мапи која може бити различитих величина и облика. Мапа је често постављена као лавиринт с различитим препрекама и рупама.
- **Отвор** (*Hole*) - Циљ: Циљ игре је поставити блок тако да он вертикално падне у одређени отвор на мапи.
- **Препреке** (*Obstacles*): На мапи се могу налазити различите препреке које отежавају кретање блока. То могу бити пропадајући блокови, рупе, мостови и друге препреке.
- **Кретање** (*Movement*): Играч користи контроле да би померао блок по мапи. Блок се може кретати напред, назад, лево и десно, и може пасти с ивица мапе ако се не постави на одговарајући начин.
- **Поновно покушавање** (*Restart*): Играч може поново покушати ниво ако не успе да постави блок у отвор. Ово је често потребно јер је игра изазовна и захтева експериментисање.
- **Нивои** (*Levels*): Игра обично има различите нивое са све сложенијим мапама и задацима. Играч напредује кроз нивое како их завршава.
- **Време** (*Time*): У неким верзијама игре *Bloxorz* може постојати временско ограничење за сваки ниво, што додатно повећава изазов.
- **Поени** (*Points*): Играчи могу освајати поене на основу брзине и ефикасности којом завршавају нивое.

- **Рестартовање нивоа** (*Reset*): Ако играч застаје или жели поновно да покуша ниво из почетка, може да користи опцију за ресетовање нивоа.

## 2.3 Популарност игре и њен настанак

Игра је настала као резултат рада Дамијана Кларка[4], талентованог програмера и дизајнера игара из Аустралије. Његова игра је први пут објављена 2007. године на популарној играчкој платформи Миниклип (*Miniclip*) [5], али пут од идеје до успешног онлајн наслова је фасцинантан. Дамијен Кларк је био инспирисан да створи *Bloxorz* једноставном идејом: поставити играча пред изазов померања блока према одређеном циљу. Овај основни концепт је био једноставан, али изузетно ефикасан. Кларк је желео да створи игру која би била интуитивна за играње, али истовремено пружила дубок и изазован доживљај.

Развој игре захтевао је много труда и пажње усмерене према детаљима. Кларк је морао да дизајнира различите нивое који би постајали све тежи како би задржали играче заинтересованим. Осим тога, морао је да размисли о физици блока, како ће се понашати и реаговати на различите површине и препреке. Ово је захтевало прецизно програмирање како би се симулирала реалистична интеракција између блока и околине.

Један од кључних фактора који је допринео популарности *Bloxorz-a* била је његова онлајн доступност. Кларк је одлучио да игру понуди бесплатно на Миниклип платформи, омогућавајући играчима широм света да је играју без икаквих трошкова. То је омогућило игри да створи широку и верну публику. Када је игра први пут објављена, привукла је пажњу играча њеном једноставношћу и заводљивим изазовима. Многи играчи су проводили сате покушавајући да савладају нивое и размењивали своје стратегије са другима. Ово је допринело стварању заједнице око игре, где су се играчи такмичили и делили савете.

Након првобитног успеха, *Bloxorz* је доживео бројне варијације и верзије које су развијене током година. С обзиром на концепт и начин на који игра доприноси развоју логике и просторног сналажења она је убрзо постала предмет многобројних истраживања из различитих области које укључују когнитивну психологију, образовање и развој вештина решавања проблема.

Једно од таквих истраживања фокусира се на проучавање когнитивних вештина. Игра захтева од играча да размишљају логички, планирају своје кораке унапред и решавају комплексне проблеме на путу ка циљу. Овај аспект игре омогућио је истраживачима да испитају процесе као што су просторна перцепција, планирање, аналитичко размишљање и способност решавања проблема код играча [6]. Истраживачи такође користе игре овог типа за испитивање стратегија везаних за решавање различитих проблема. Анализом како играчи приступају изазовима у игри, они могу боље разумети различите стратегије које људи примењују при решавању проблема и како се те стратегије развијају током времена.

Осим тога, игре као што је *Bloxorz* користе се и као алати за учење програмирања. Концепти као што су алгоритми и логичке инструкције, који су кључни за програмирање, могу се боље разумети путем интерактивних игара овог типа. То је посебно корисно у образовном контексту, где се игре користе како би се подучавале основе програмирања у школама.

Употреба функционалне магнетске резонанце (fMRI) или електроенцефалографије (EEG) током играња игара као што је *Bloxorz* омогућава истраживачима да проуче како мозак реагује на различите врсте изазова и активира различите делове мозга током решавања проблема [7].

На крају, игре попут *Bloxorz-a* могу бити корисни образовни алати за развој просторних вештина код деце. Њихова интерактивна природа може учинити учење и сналажење у простору забавнијим и ангажованијим [8] .

Иако игра *Bloxorz* није првобитно створена у сврху истраживања, њен једноставан, али изазован концепт и механика представљају богат извор информација и могућности за различите области истраживања и образовања.

## 2.4 Постојећа решења

*Bloxorz* је првобитно развијен у *Adobe Flash* технологији [9]. *Flash* је био популарна платформа за веб игре и анимације током већег дела 2000-их и раних 2010-их. *Adobe Flash* је омогућавао програмерима да креирају различите интерактивне и мултимедијалне садржаје, укључујући и саме игре. Програмски језик који се често користио за развој игара у *Adobe Flash-y* био је *ActionScript* [10]. *ActionScript* је језик за скриптирање који је био сличан *JavaScript-y* и користио се за програмирање логике игре, интеракцију са корисником и анимацију унутар *Flash* апликација. Међутим, важно је напоменути да је *Adobe Flash* застарела технологија која је престала да буде подржана и изгубила популарност током 2010-их година. Ова технологија није била ефикасна у искоришћавању ресурса рачунара и није радила добро на мобилним уређајима. Након што је Adobe престао са подршком за Flash, ова верзија игре постала је неприступачна на већини модерних веб прегледача.

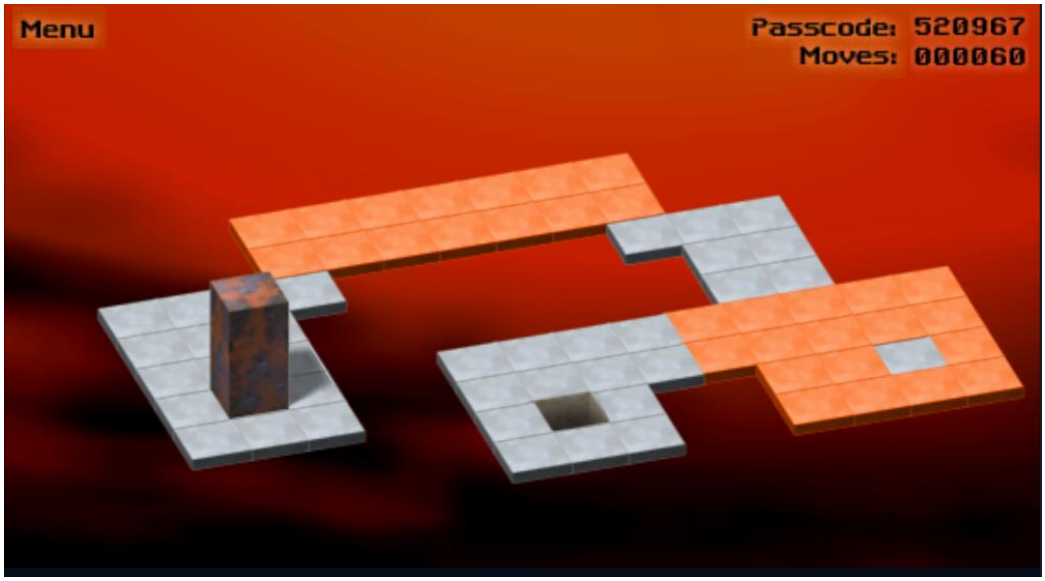
*Creat Studios* је објавио 8. јануара 2009. године видео игру под називом *Cuboid* [11] која је била прилагођена за играње на PlayStation 3. Концепт игре сличан је *Bloxorz-y* осим што нивои садрже нове елементе са којима се мора комуницирати, као што су прекидачи, телепортери и дрвене платформе. Прекидачи на одређеним нивоима морају бити активирани да би се напредовало кроз ниво. Постоје посебни прекидачи чијим се активирањем приказују скривени делови терена. Постоје и појачања (*power ups*), укључујући она која деле блок на два дела, и она која омогућавају играчу да направи још неколико покрета на нивоима са ограниченим бројем покрета по нивоу. *Cuboid* је добио оцену 79/100 на *Metacritic*-у [12] на основу 13 рецензија, што указује на повољне критике.

Верзија *Bloxorz: Roll the Block* из 2018. године донела је нову графику и модернији дизајн сцене. Ова верзија је доступна на различитим платформама, укључујући и мобилне уређаје.

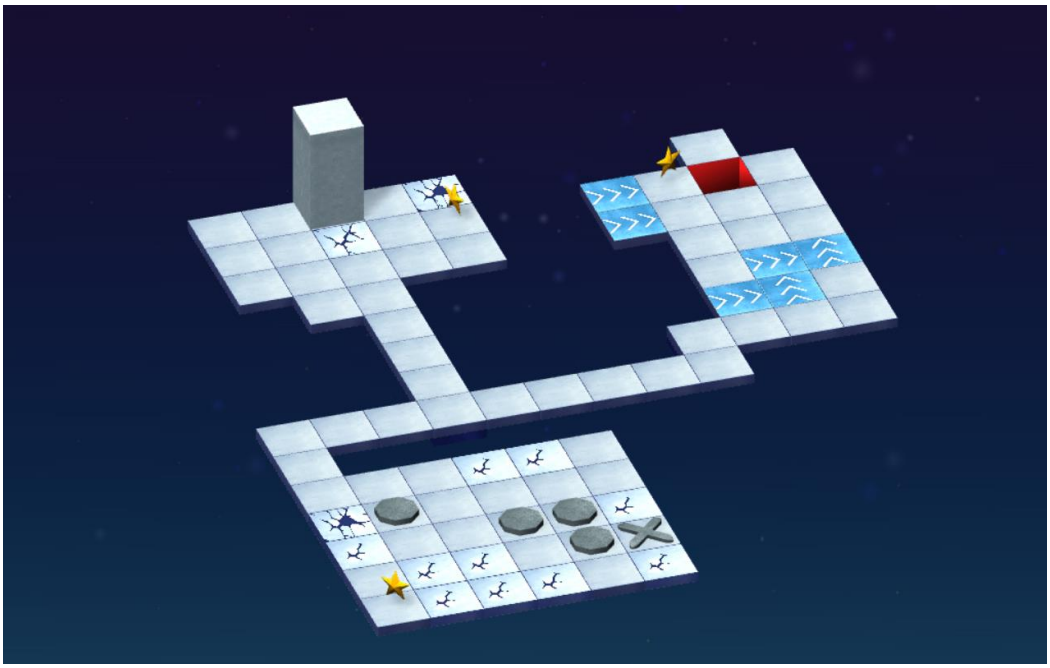
*Миниклип* је објавио 3. августа 2022. године *Bloxorz 2*, нову верзију игре која се састоји од оригиналне Кларкове игре и његових нивоа (Слика 2.4.1). и новог додатка, односно, колекције нивоа под именом *Winter* (Слика 2.4.2). Осим што игра има нови дизајн и инспирисана је зимом она садржи нова поља од леда која пуцају при контакту са било којим делом блока. У зависности од напрслости поља, постоје она која се могу искористити једном, два или три пута што доприноси томе да играч мора пажљиво да промисли своју тактику и наредне потезе пре него што их искористи. Нова врста препреке у виду клизавих поља

преусмерава блок на суседно поље уколико се он нађе усправно на истом. Смер клизавости приказује се у виду стрелица на самом пољу.

Током времена, а посебно након гашења *Flash* платформе велики фанови направили су многобројне верзије ове игре. Њихово стварање постало је могуће завађујући напретку у развоју софтвера и технологији. Иновативни развојни тимови су непрестано радили на побољшањима и додавању нових елемената игре како би привукли већи број играча и прилагодили игру различитим платформама. Свака нова верзија често је донела унапређења графике, механике игре или додатне садржаје који обогаћују искуство играча. Због тога се данас на Интернету и мобилним уређајима може наћи велики број верзија ове игре.




***Слика 2.4.1 – Изглед 4. нивоа оригиналне игре Bloxorz***



***Слика 2.4.2 – Изглед 3. нивоа игре Bloxorz 2***

# 3. ТЕХНИЧКИ АСПЕКТИ И ЗАХТЕВИ ПРОЈЕКТА

Ово поглавље обухвата упознавање са функционалним програмирањем, *Scala* програмским језиком, као и *ScalaFX* библиотеком која је коришћена за развој пројекта. Крај поглавља фокусира на техничке захтеве приликом имплементације 2Д игре и преглед аспеката који су били битни при креирању 3Д модела.

## 3.1. Програмски језик и коришћени алати

### *3.1.1. Функционално програмирање*

Настанак функционалног програмирања (*Functional Programming*) [13] обично се повезује са потребама и истраживањима из области математике и рачунарске науке. Његови корени леже у ламбда рачуну, формалном систему развијеном током 1930-их ради проучавања дефиниције и примене функција и рекурзије. Главна мотивација за коришћење ламбда рачуна лежи у томе што он укључује упрошћења која семантику чине једноставном и интуитивном [14]. Његове две кључне карактеристике јесу:

- Ламбда рачун функције третира анонимно - без давања имена.

На пример, уместо функције *sum(x,y) => x+y* може се користити њена анонимна форма *(x,y) => x+y*

- Ламбда рачун користи само функције са једним улазом

Наиме, уместо функције која прима више параметара може се написати функција која прима један параметар и као повратну вредност враћа функцију која такође прима један параметар. На пример, функција која рачуна збир квадрата два броја *(x,y) => x * x + y * y* се коришћењем ламбда израза представља као функција *(x) => ( y => x * x + y * y )( y )*

Као што смо видели на претходним примерима ламбда функција, функционално програмирање је парадигма програмирања која се фокусира на рад са функцијама као основним јединицама за развој софтвера. У функционалном програмирању, функције су третиране као првокласни објекти, што значи да се могу креирати, складиштити у промењивима, проследити као аргументи другим функцијама и вратити из функција као резултати.

Кључне особине функционалног програмирања укључују:

- **Функције вишег реда:** Представљају функције које могу примати друге функције као аргументе или их враћати као резултате. Ово омогућава кратак и изразит код,

повећава апстракцију и доприноси писању истих функционалности са значајно мањим бројем линија кода у односу на нефункционалне језике.

- **Чисте функције:** Функционални програмски језици теже коришћењу чистих функција, што значи да свака функција добија улазне параметре и враћа резултат, без небезбедних спољних утицаја (споредних ефеката) као што су мутације или и/или операције.
- **Имутабилност:** Подаци се третирају као имутабилни (непромењиви), што значи да се не мењају након што се креирају. Сваки пут када се врше одређене измене над подацима креирају се нове верзије података. Мана овог приступа јесте што захтевају обимно копирање уместо једноставног ажурирања.
- **Рекурзија:** Уместо циклуса, функционално програмирање често користи рекурзију за обраду података и решавање проблема.

Популарни функционални програмски језици укључују *Haskell*, *Lisp, ML*, *Scala*, и *Clojure*. Функционално програмирање се користи за решавање различитих врста задатака, укључујући анализу података, веб развој, и високоперформантно рачунање.

### *3.1.2. Програмски језик Scala и развој 3Д апликације коршћењем ScalaFX библиотеке*

За израду мастер рада коришћен је програмски језик *Scala* [15]. *Scala* је вишенаменски програмски језик који је изворно развијен са циљем да комбинује карактеристике објектно оријентисаних језика са функционалним програмирањем. Она подржава објектно оријентисано програмирање, где се сви ентитети (класе, објекти, методе) представљају као објекти док са друге стране има јак акценат на функционалном програмирању, омогућавајући програмеру да функције користи као праве објекте, лако имплементира и користи анонимне функције и ради са угњежденим функцијама и класама.

За развој 3Д компонената игре коришћена је *scalaFX* [16] библиотека која представља скуп пакета за графику, мултимедију и веб. Она је уско је повезана са програмским језиком Java и стога се једноставно могу користити све постојеће услужне опције из скупа пакета програмског језика Јава. Поред менија, дугмади, текстуалних поља, мултимедије и било које друге компоненте графичког интерфејса прозора постоји колекција 2Д објеката, који се могу стилизовати помоћу *css-a* (*Cascading Style Sheets*). То јест, *scalaFX* и *javaFX* [17] користе исти стандард који се обично користи за веб дизајн. *ScalaFX* се такође може користити за графички дизајн модерних видео игара и других апликација које захтевају визуелизацију и анимације у 3Д. Постоји велика колекција услужних програма за креирање облика, прелаза, перспектива, покрета и других неопходних артифаката. Његова употреба захтева одређено техничко знање, али – као и увек у свету отвореног кода – на Интернету се може наћи много информација и туторијала.

Скоро свака *ScalaFX* апликација заснива се на следећим концептима:

- Апликација има најмање једну фазу, која представља GUI (*Graphical User Interface*) прозор, и сцену која је контејнер за графичке елементе.
- У прозор се може додати хијерархија чворова или контрола. То могу бити, на пример, компоненте попут панела, табела или мрежа, или компоненте интерфејса као што су дугмад, оквири за текст, менији и клизачи.
- Интеракција са корисником обрађује уз помоћ догађаја (*Event*), који могу бити на пример кликови мишем или радње на тастатури.

За потребе адекватног прегледа терена и лакше праћење и позиционирања главног актера у 3Д простору неопходно је било имплементирати камеру са перспективном пројекцијом. За потребе едитора за прављење новог нивоа имплементирана је и паралелна ортографска камера. У наставку су описане основне карактеристике поменутих камера. Оне објашњавају потребу за имплементацијом обе врсте.

Што се тиче камере са перспективном пројекцијом она користи конвергентне зраке светла како би створила 2D слику која симулира начин на који људско око види свет око себе. Приликом гледања кроз перспективну камеру, објекти који су ближи камери изгледају веће, док објекти удаљени од камере изгледају мање. Перспективна камера производи реалистичне слике које боље одражавају наше варање стварног света. Ова врста камере се често користи у фотографији, филмским производњама, видео играма и другим применама где је важно створити 2D или 3D слике које одговарају људском визуелном искуству.

Са друге стране, паралелна ортографска камера користи паралелне линије за пројекцију светла на сензор камере. Сви зраци светла долазе под правим углом на сензор камере, што значи да нема перспективних ефеката. Слике снимљене паралелном камером често изгледају као да су равне и без дубине, сличне техничким цртежима или цртежима у компјутерској графици. Паралелне камере се често користе за ортографске пројекције у рачунарској графици, инжењерству, архитектури и техничким применама где је важна прецизност и одржавање пропорција.

Укратко, паралелна камера се користи када треба одржати пропорције и димензије објеката, док се перспективна камера користи када је потребно постићи осећај дубине и реалистичнији приказ сцене [18].

За имплементацију овог пројекта коришћено је *jdk-16.0.2* јавино окружење за развој софтвера док је пројекат је реализован у *IntelliJ IDEA* [19] развојном окружењу.

## 3.2. Технички захтеви за имплементацију игре

Проблем који се јавља приликом развоја видео-игре јесте осмишљавање њеног концепта. На основу ставки и захтеваних функционалности које игра треба да испуни да би од пројекта димензије лабораторијске вежбе била надограђена до мастер рада, треба осмислити причу и ток игре тако да она буде што занимљивија, интуитивнија, кориснички оријентисана и да држи пажњу играча на дуже стазе. У наставку су наведене основне функционалности које је игра требало да садржи. У сврху вежбе, примене функционалног програмирања и како бисмо обогатили и унапредили концепт, уведене су додатне

функционалности. Додатне опције укључују могућност креирања нових нивоа, играње претходно креираног нивоа и нивоа који се учитава из фајла и аутоматско решавање игре.

Основне функционалности подразумевају:

- Учитавање мапе терена из фајла. Мапа терена је матрица која може да садржи следеће симболе:
  - симбол “o” који представља обичну плочу;
  - симбол “–” који означава да на датој позицији није постављена плоча;
  - симбол “C” који означава стартну позицију;
  - симбол “T” који означава крајњу позицију;
  - симбол “.” који означава специјалну плочу – уколико се блок постави на ову плочу у усправном положају, плоча нестаје и блок пропада ван терена.
- Започињање нове игре избором неке од расположивих мапа.
- Одигравање потеза. Одигравање представља померање блока у зависности од потеза који корисник зада (доле, горе, лево, десно), уз приказ стања игре након одигравања потеза. Уколико се након одигравања блок доведе у циљну позицију, игра се завршава победом. Уколико се након одигравања блок избаци ван терена, игра се завршава поразом.
- Одигравање секвенце потеза учитавањем из фајла. Секвенца потеза садржи редом ознаке потеза, сваки у засебном реду. Ознаке потеза су:
  - доле: “down”
  - горе: “up”
  - лево: “left”
  - десно: “right”

Креирање нових мапа на основу постојећих:

- Уклањање задате плоче са ивице терена.
- Додавање плоче на задату позицију на ивици терена.
- Замена обичне плоче на задатој позицији специјалном.
- Замена специјалне плоче на задатој позицији обичном.
- Постављање стартне позиције на задато поље, при чему се оригинална стартна позиција мења обичном плочом.
- Постављање циљне позиције на задато поље, при чему се оригинална циљна позиција мења обичном плочом.

Предвиђене су следеће предефинисане операције едитора мапе:

- Инверзија: резултујућа мапа се добија тако што се на оригиналној мапи замене стартна и циљна позиција.
- Замена: резултујућа мапа се добија тако што се на оригиналној мапи све специјалне плоче претварају у обичне плоче.
- Филтрирање: уколико је макар један од суседа дате плоче на вертикалној или хоризонталној раздаљини мањој или једнакој N специјална плоча, заменити дату плочу обичном плочом.
- Испис решења игре у облику секвенце потеза који доводе блок из почетне у крајњу позицију.

Након дефинисања основних захтева, њихове имплементације и имплементације опције за креирање нових мапа, неопходно је апликацију прилагодити 3Д моделу. Ставке које је било потребно размотрити пре овог дела имплементације јесу:

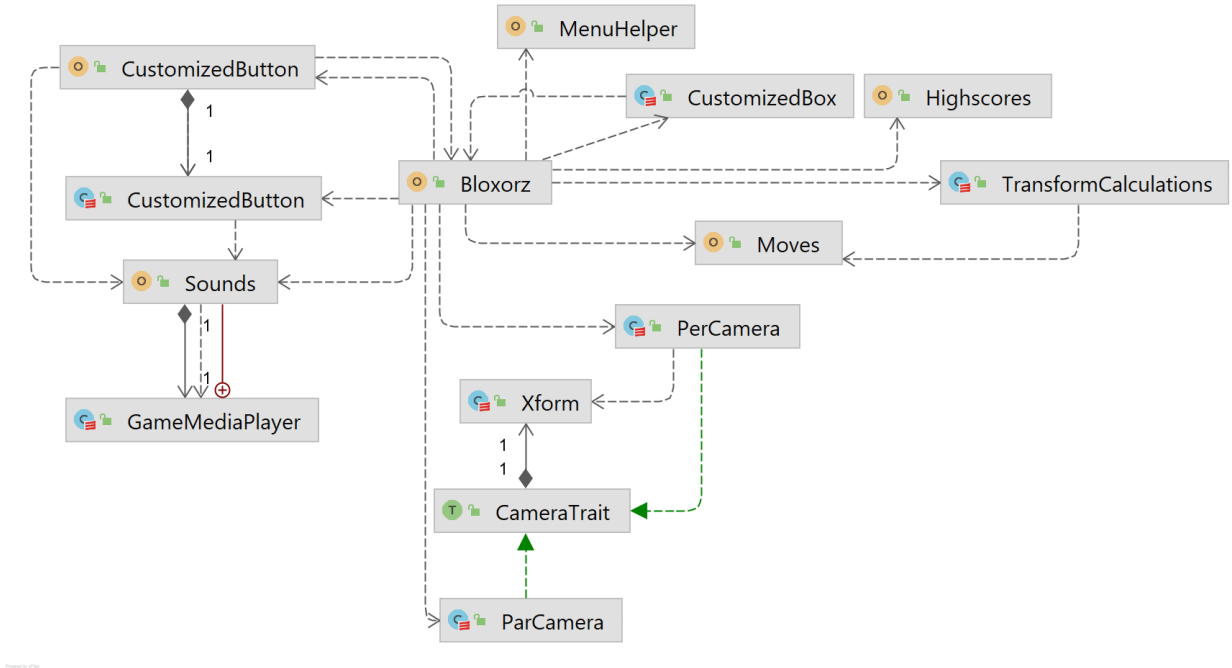
- Превођење 2Д у 3Д графику – уместо правоугаоника који су коришћени у случају 2Д графике неопходно је било имплементирати 3Д сцену, игру састављену од *Box* објеката, затим додати камеру и осветљење
- Осмишљавање полигона
- Развој корисничког интерфејса и приказ тренутног стања игре – распоред објеката на сцени током игре
- Убацивање менија за лакшу комуникацију са апликацијом
- Осмишљавање начина рангирања и приказ ранг-листе
- Додавање музике и звучних ефеката
- Паузирање игре – подразумева тренутно заустављање игре и накнадни наставак са истим стањем игре као у тренутку заустављања
- Осмишљавање начина на који ће корисник на што интуитивнији начин интераговати са едитором и његова имплементација
- Снимање креираних нивоа и ранг листе у текстуалне фајлове ради лакшег уређивања

# 4. Имплементација игре

У наставку овог поглавља биће дат преглед архитектуре система и укратко ће бити описане најважније класе, објекти и црте. Након тога, у другим делу поглавља, биће објашњени главни детаљи имплементације.

## 4.1. Пројекат софтвера

Дијаграм и начини на који су основне класе, објекти и црте у пројекту повезани дат је на слици 4.1.1.




***Слика 4.1.1 – UML дијаграм основних класа, објеката и црта игре Bloxorz***

Класа `Bloxorz` садржи главне функције апликације за конфигурацију и контролу тока игре. Она садржи објекат `Blox` у коме су дефинисани сви типови поља који се могу наћи у игри, а то су `SPECIAL`, `BASIC`, `EMPTY`, `EMPTY2`, `START` и `END` ( `EMPTY` се користи за потребе тренинг и такмичарског мода јер је прозирне боје док се `EMPTY2` користи приликом креирања нивоа). Тренутно стање игре памти се и рачуна уз помоћ `case` класе `State` која приликом креирања прима параметре попут тренутног нивоа, тренутне позиције блока, стање игре и распоред поља по терену. За рачунање новог стања користи се функција `newState` `State` класе која као повратну вредност враћа нови објекат `State` класе. Она прима

параметар типа `Moves` који представља екстензију `scala.Enumeration` класе и има четири предефинисане вредности `Left`,`Right`,`Down` и `Up` које представљају смер кретања блока. На основу тренутног стања и одиграног потеза рачуна се нови статус игре који за потребе даље обраде може имати три вредности које се представљају уз помоћ `GameStatus` енумерације. То су `Game`, `Win` и `Fail` при чему `Game` представља регуларан наставак игре, `Win` победу, а `Fail` губитак, односно, уколико је блок пао са ивице терена или пропао кроз специјално поље.

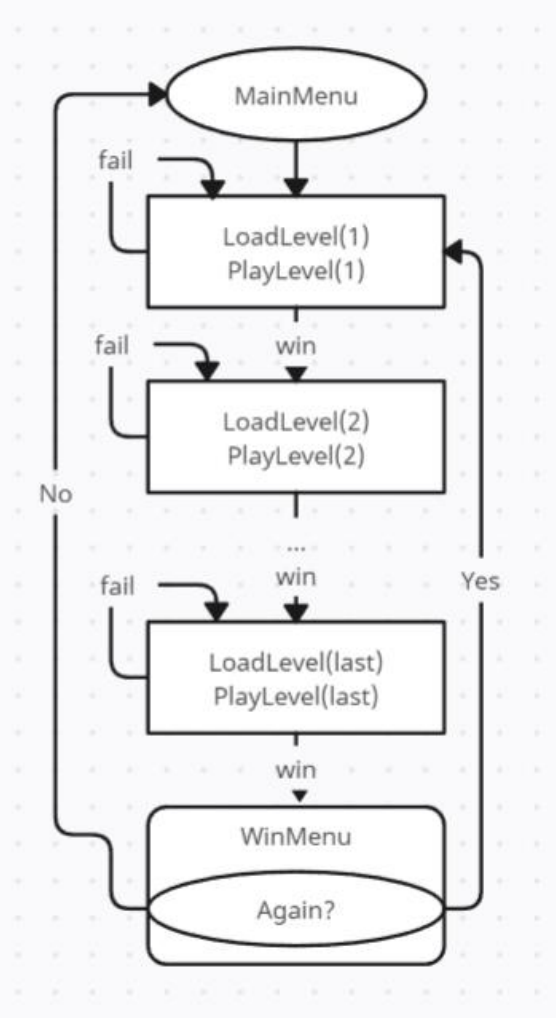
Ток игре функционише по следећем принципу:

- Одабира се жељени ниво – било путем опције за учитавање нивоа из фајла или одабиром тренинг (*Training*) или такмичарког (*Start*) мода из почетног менија.
- На основу одабраног нивоа, учитавањем из текстуалног фајла формирају се листе позиција плоча на основу њихових карактеристика и добија се почетна позиција *Blox*-а. Уз помоћ формираних листа креира се нови објекат класе `State` са почетним стањем `GameStatus.Game`.
- Након формирања новог стања, следи креирање сцене и њених елемената и њено додавање на прозор игре. Сцену чине главна – 2Д сцена са дугметом за паузирање игре и текстуалним исписом тренутног стања игре и 3Д подсцена (`SubScene`) главне сцене којој се додају сви 3Д објекти и за коју се затим везује перспективна камера.
- Следи додавање `KeyEvent`-а на новокреирану сцену који подразумева активирање звучних ефеката, повећање броја одиграних потеза, ажурирање новог стања позивом функције `newState State` класе и прослеђивањем одиграног потеза, затим ажурирање садржаја 3Д сцене. Након ажурирања стања, у оквиру догађаја се проверава и да ли је ново стање можда `Win` или `Fail`. У тим случајевима креира се адекватна анимација.
- Након завршене анимације, у случају тренинг мода следи креирање нове сцене и менија паузе (Слика 4.1.3). На крају такмичарског мода и успешно пређеног нивоа покреће се следећи ниво (*level + 1*). Супротно, у случају пада са ивице поново се покреће исти ниво (Слика 4.1.2).
- Уколико корисник кроз мени паузе одабере опцију за решавање нивоа из фајла, креира се нова нит и унутар ње робот из јавиног `awt.Robot` пакета симулира одигравање секвенце потеза у размацима од 200 милисекунди.

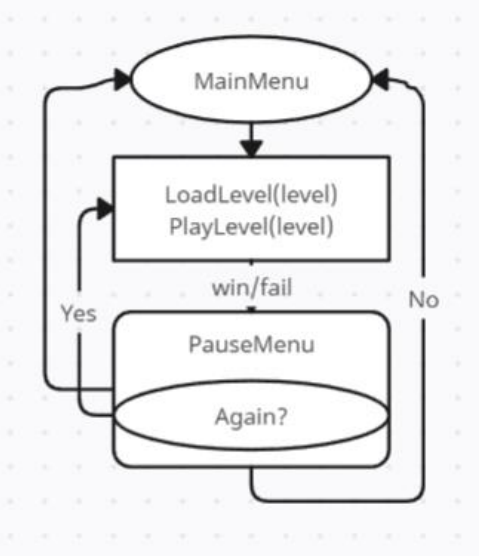
Што се тиче креирања новог нивоа, оно функционише по сличном принципу:

- Одабира се жељени ниво – одабиром једног од четири предефинисана тренинг нивоа из менија.
- На основу одабраног нивоа, учитавањем из текстуалног фајла формира се листа 3Д објеката.
- Креира се хијерархија – формира се главна 2Д сцена са дугметом за напуштање едитора и дугметом за чување мапе. Затим се додаје 3Д сцена са 3Д објектима и паралелном камером.

- За сваки од 3Д објеката, у зависности од његових карактеристика креира се одређени скуп операција које ће се приказати услед клика мишем на посматрани објекат. Затим се оне додељују објектима. Функције углавном подразумевају промену материјала посматраног поља у складу са траженим захтевима.
- На крају, приликом чувања мапе, листа 3Д објеката мапира се у листу знакова у складу са материјалима објеката и уписује се у одабрани текстуални фајл.




***Слика 4.1.2 – Ток игре такмичарског нивоа***




***Слика 4.1.3 – Ток игре такмичарског нивоа***

Објекат `MenuHelper` садржи 2Д елементе типа `Text` и `TextField` који се користе приликом прављења менија и 2Д елемената графичког корисничког интерфејса.

`Case` класа `HighScores` задужена је за дохватање и упис најбољих резултата у текстуалне фајлове.

Класа/објекат `GameMediaPlayer` служи за креирање, складиштење и манипулисање музиком и звучним ефектима.

За потребе праћења кретања блока по полигону и прегледа 3Д сцене неопходно је било имплементирати додатне класе које проширују класе `PerspectiveCamera` и `ParallelCamera` из скалиног `scalafx.scene` пакета. Ове класе смешене су у `Cameras.scala` фајлу и садрже подршку за паралелну ортографску камеру и камеру перспективном пројекцијом. Камера са перспективном пројекцијом коришћена је за приказ сцене током игре док је паралелна камера коришћена за преглед терена у случају опције за прављење новог нивоа.

## 4.2. Имплементација

У овом поглављу биће описано пар детаља имплементације као и техничке карактеристике развијеног софтвера.

*Tailrec* је скраћеница која се користи да означи репну рекурзију [20]. Рекурзија је техника у програмирању где функција позива саму себе како би решила одређени проблем. Код репне рекурзије, последња операција у функцији је позив исте те функције. То значи да нема операција после рекурзивног позива, што омогућава оптимизацију у неким програмским језицима. Када се користи репна рекурзија, компајлер или интерпретер може да изведе оптимизацију како не би морао да додаје нове нови простор за параметре (стек фраме)

на стек током сваког рекурзивног позива, већ да модификује и поново искористи постојећи простор.

Ово је значајно јер класична рекурзија може довести до прекорачења стека (*stack overflow*) ако није пажљиво направљена, посебно кад постоји велики број рекурзивних позива. Репна рекурзија омогућава избегавање тог проблема јер не ствара нове инстанце функције на стеку.

У суштини, репна рекурзија је техника која омогућава ефикасно решавање проблема помоћу рекурзивних позива тако да се избегну могући проблеми с прекорачењем стека. У наставку је дат део кода и начин на који је имплементирано решавање нивоа (Табела 4.2.1).

```
@tailrec
def writeToFile(current: Int, state: List[(Double, Double, Double, Double)], accFrom: List[Int]): Unit
= {
  if (current == 0) writer.close()
  else {
    if (state(accFrom(current)). 1 < state(current). 1) writer.write("down\n")
    else if (state(accFrom(current))._1 > state(current)._1) writer.write("up\n")
    else if (state(accFrom(current))._2 < state(current)._2) writer.write("left\n")
    else if (state(accFrom(current))._2 > state(current)._2) writer.write("right\n")
    writeToFile(accFrom(current), state, accFrom)
  }
}

@tailrec
def solveLevel(current: Int, state: List[(Double, Double, Double, Double)], accFrom: List[Int], end:
(Double, Double)): Unit = {
  if (state.length < current + 1) {
    writer.write("This one has no solution")
    writer.close()
  }
  else if (state(current). 1 == state(current). 3 && state(current)._2 == state(current)._4 &&
state(current). 1 == end. 1 && state(current)._2 == end._2) {
    writeToFile(current, state, accFrom)
  }
  else {
    val nxt = nextPosition(state(current)).filterNot(p => state.contains(p))
    solveLevel(current + 1, state ::: nxt, accFrom ::: List.fill(nxt.length)(current), end)
  }
}


solveLevel(0, List((end._1, end._2, end._1, end._2)), List(0), blox.head)
```

***Табела 4.2.1 – Имплементација функције за решавања нивоа***

Функција `solveLevel` као аргументе прима тренутну позицију на којој се функција налази приликом рекурзивне обраде свих стања у којима блок може да се нађе. Стања се допуњавају и памте кроз аргумент `state` који представља листу од четири `Double` вредности. Оне означавају x и y координате оба дела блока (пошто је блок димензија 1x1x2). Даље се користе акумулатор у коме се памти из које позиције претходно је блок доспео у тренутну позицију и аргумент `end` који представља крајњу позицију у коју блок треба да доспе. Први услов за заустављање рекурзије јесте да је функција обрадила сва стања у којима блок може да се нађе и да није дошла до решења. У том случају у одабрани текстуални фајл исписује се одговарајућа порука „`This one has no solution`“. У другом случају када је решење пронађено позива се функција која на основу акумулатора стања и акумулатора позиција из којих се до тих стања стигло рекурзивно реконструише путању.

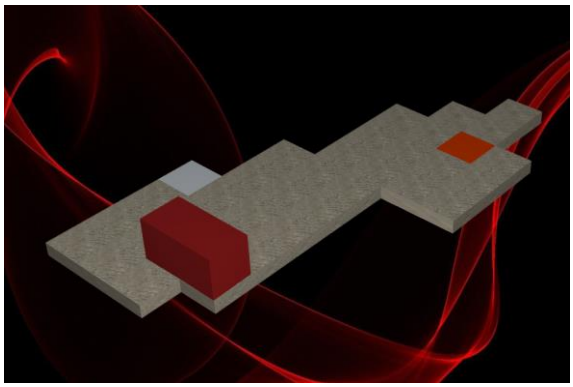
Последњи (репно рекурзивни) део функције користи функцију `nextPosition` која на основу распореда поља на терену (у зависности од нивоа) и тренутне позиције враћа листу позиција у којима се блок може наћи у следећем потезу. Затим се та листа филтрира и у `state` акумулатор додају само оне позиције у којима се блок није већ нашао. Након тога функција се позива за следећи елемент `state` акумулатора.

С обзиром да се у акумулатору памти из које позиције се доспело у тренутну, тражење решења у коду одређује се у правцу од циља ка старту да би испис ишао у супротном правцу.

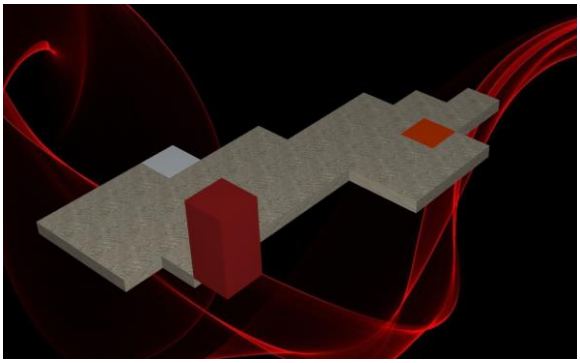
Следећи део имплементације којим ћемо се позабавити у наставку јесу анимације. Да би се кориснику на адекватан начин приказало које су последице сваког одиграног потеза а посебно оних који су довели до победе, односно губитка, било је неопходно направити визуелне ефекте на основу којих би корисник јасно могао да схвати шта је била грешка. За те потребе додате су анимације које приближно симулирају гравитационе силе и физику објекта.

Наиме, анимације су подељене у 3 основновна типа. За потребе лакшег разликовања назваћемо их једноставна транслација, директна ротација и ротација на страну.

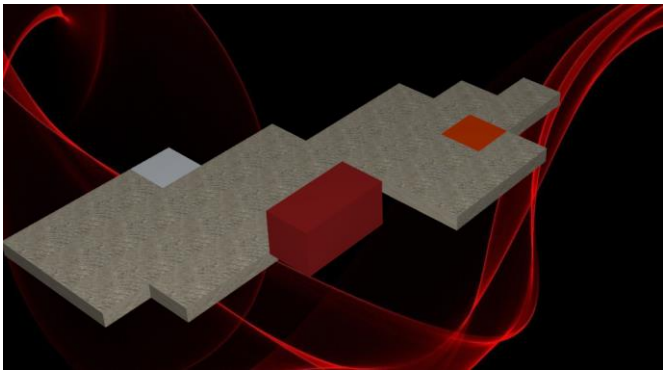
- **Једноставна транслација** – овај тип анимације је неопходан када је блок позициониран усправно а ефекат који је потребно применити јесте једноставна транслација у правцу *z* осе. Ова анимација користи се у случајевима када се блок нађе усправно на циљу (победа), на блоку који пропада или је усправно пао са ивице терена. На слици 4.2.1 и слици 4.2.2 дат је пример у коме се користи ова транслација приликом усправног пада са ивице терена. Једноставна транслација користи се и у случају када је блок водоравно пао са терена целом својом дужином (Слика 4.2.3).



***Слика 4.2.1 – Стање игре пре усправног пада са ивице терена***

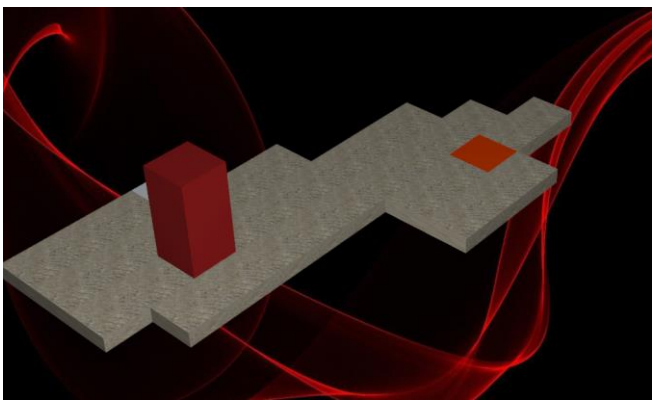


***Слика 4.2.2 - Стање током пада са ивице терена, ситуација у којој се примењује једноставна транслација на усправно постављен блок***

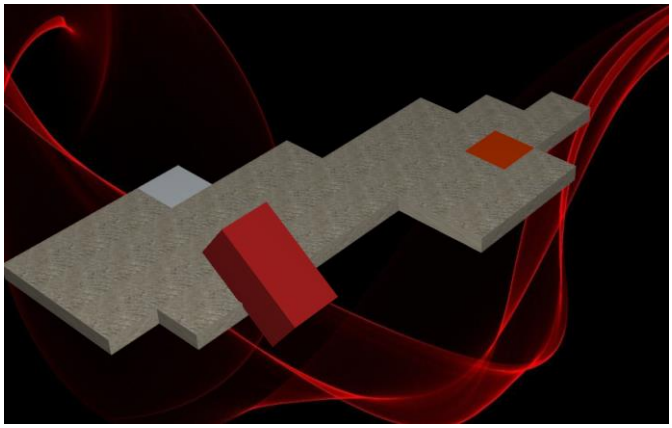


***Слика 4.2.3 - Стање игре током пада са ивице терена, ситуација у којој се примењује једноставна транслација на водоравно постављен блок***

- **Директна ротација** – овај тип анимације јавља се уколико је блок из усправног стања доспео у стање у коме је једна његова половина изнад провалије (Слика 4.2.4 и Слика 4.2.5). Због ротационе силе [21] која делује на блок приликом кретања, он ће приликом погрешног потеза наставити са својом ротацијом и пасти у провалију. Овај ефекат имплементиран је уз помоћ транзиција из `scalafx.animation` пакета. Прво се направи одговарајућа паралелна транзиција која симулира ротацију преко ивице терена уз адекватно транслаторно кретање тежишта блока тако да он изгледа као да клизи преко ивице. Након завршене анимације додаје се транслација која симулира наставак пада.

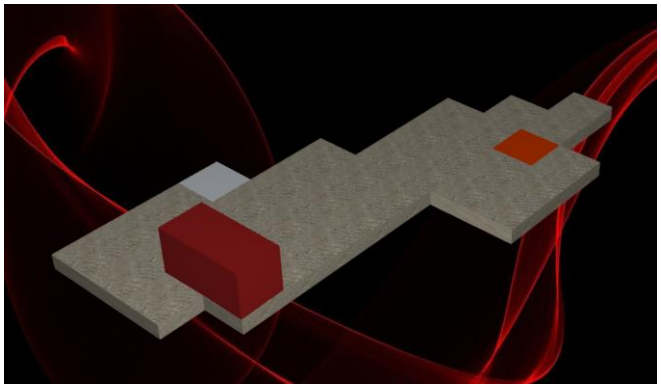


***Слика 4.2.4 – Стање игре пре пада са ивице терена***

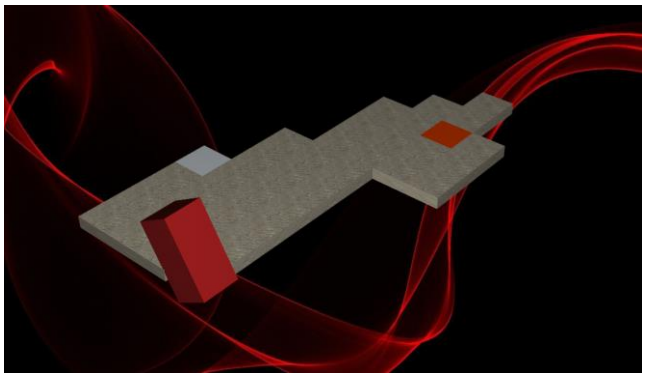


*Слика 4.2.5 – Стање игре током пада блока са терена, примена директне ротације*

- **Ротација на страну** – овај тип анимације подсећа на претходни случај осим што се јавља уколико је блок из водоравног стања доспео у стање у коме је једна његова половина изнад провалије (Слика 4.2.6 и Слика 4.2.7). У овом случају се због нестабилности [22] он преврће и ротира на страну која је изнад провалије.



*Слика 4.2.6 – Стање игре пре пада са ивице терена*



*Слика 4.2.5 – Стање игре током пада блока са терена, примена ротације на страну*

Функција која креира неопходне анимације има следећу декларацију (Табела 4.2.2):

```
def calculate(direction: Moves, blox: List[(Double, Double)], emptyBloxList: List[(Double, Double)],
bloxBox: Box): Seq[Transition]
```

***Табела 4.2.2 – Декларација методе за креирање анимација***

Прорачуни везани за смер ротације и транслације, врше се на основу крајњег стања блока које прослеђује кроз параметар `blox`, затим смера у ком се блок кретао (`direction`) и листе празних блокова која служи да би се установило којим делом блок не додирује терен у посматраном тренутку. Параметар `bloxBox` представља 3Д објекат типа `Box` из `scalafx.scene.shape` пакета који представља објекат блока који се у датом тренутку налази у сцени и над којим је потребно извршити анимације.

## 4.3. Техничке карактеристике

У наставку су дате техничке карактеристике апликације.

***Табела 4.3.1 – Техничке карактеристике***

| Метрика | Вредност |
|---|---|
| Број фајлова | 34 |
| Број скала класа | 9 |
| Број линија кода | 1079 |
| Величина ресурса | 2905 kB |
| Величина IntelliJ пројекта | 503 MB |
| Величина извршног(JAR) фајла | 54.9 MB |

# 5. Функционална спецификација – упутство за употребу

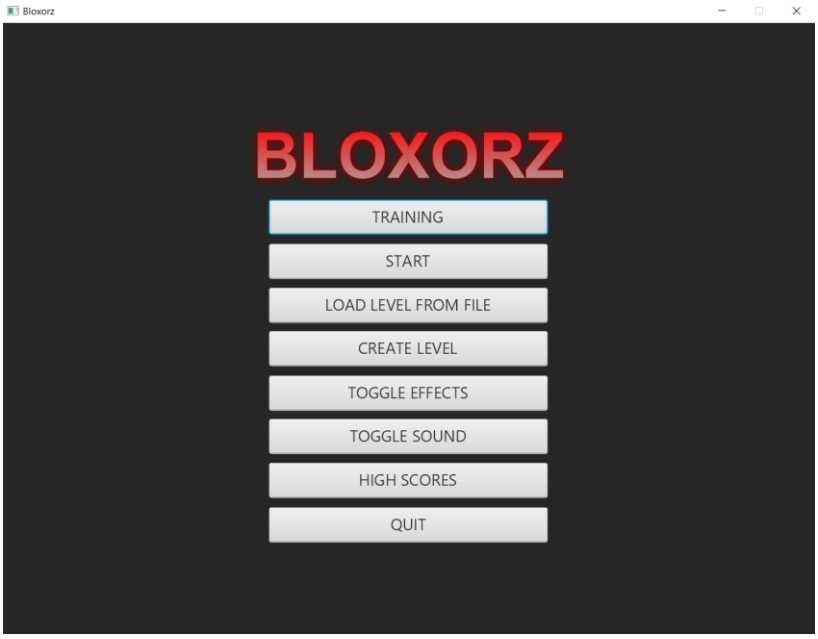
У наставку ће бити описани сви елементи који су од значаја за корисника, начини на које корисник интерагује са апликацијом. Такође, биће објашњени сви аспекти за које се могу јавити нејасноће у току игре.

## 5.1. Почетни мени

Приликом покретања апликације играчу се на екрану појављује мени (Слика 3.1) са следећим опцијама:

- TRAINING – започиње нову игру која се не бодује и служи за вежбање и упознавање са правилима, садржи једноставније нивое
- START – започиње нову игру која се бодује
- LOAD LEVEL FROM FILE – опција за учитавање нивоа из текстуалног фајла
- CREATE LEVEL – опција која омогућава играчу да креира нови полигон максималне димензије од 14 блокова
- TOGGLE EFFECTS – гашење/паљење звучних ефеката
- TOGGLE SOUND – гашење/паљење позадинске музике
- HIGH SCORES – овара листу најбољих играча
- QUIT – илаз, односно, заварање и напушање апликације

Жељену ставку из менија корисник може да одабере кликом на исту или позиционирањем фокуса уз помоћ стрелица тастатуре и одабиром кликом на тастере SPACE или ENTER.




***Слика 5.1.1 – Почетни мени***

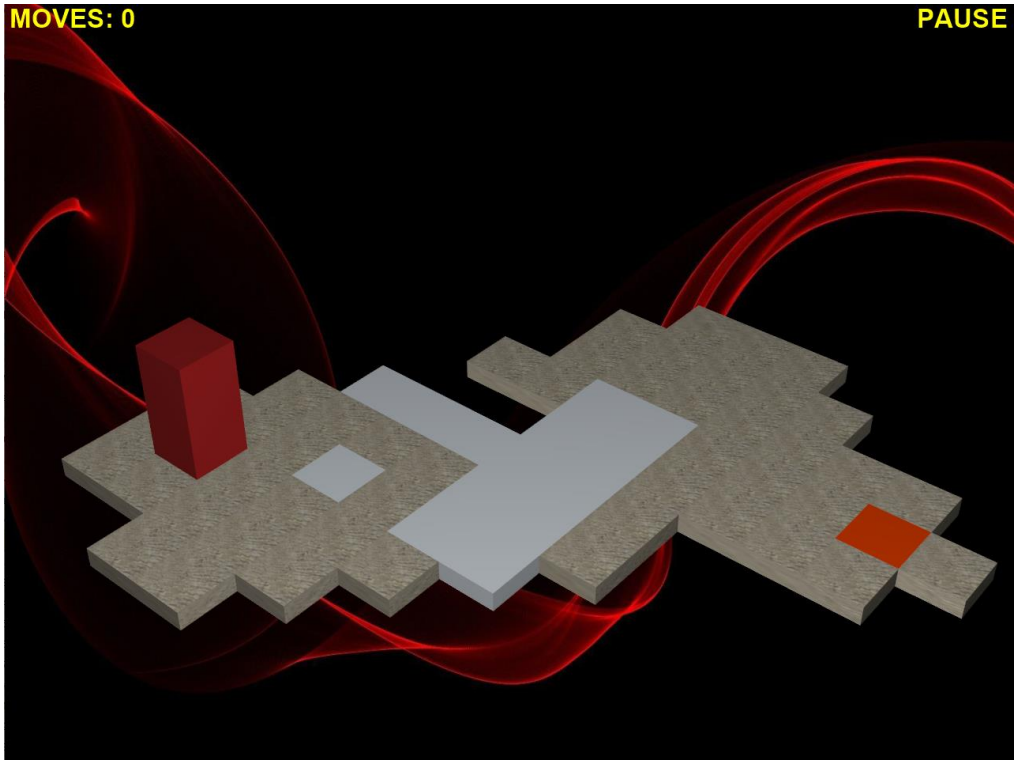
## 5.2. Играчки интерфејс

### *5.2.1. Тренинг*

Одабиром опције TRAINING, кориснику се приказује нови екран на коме може да одабере који од понуђена четири тренинг нивоа жели да игра. Након одабира жељеног нивоа, игра се покреће. На новоприказаној сцени налази се полигон кога чине:

- Обична поља – немају никакав ефекат, блок може слободно да се креће по њима
- Специјална поља – поље које не може да издржи тежину целог блока и зато уколико се блок нађе усправно на овом пољу, оно пуца и играч губи игру
- Циљ – комбинацијама превртања у сва 4 смера, играч треба да позиционира блок усправно на ово поље и тиме оствари победу
- Блок – је Блок величине 1x1x2 јединица и може се превртати у сва 4 смера

Осим полигона и блока у горњем левом углу играчког инерфејса налази се и индикатор броја одиграних потеза који у случају тренинг нивоа служи за праћење прогреса приликом вежбе (Слика 5.2.1).

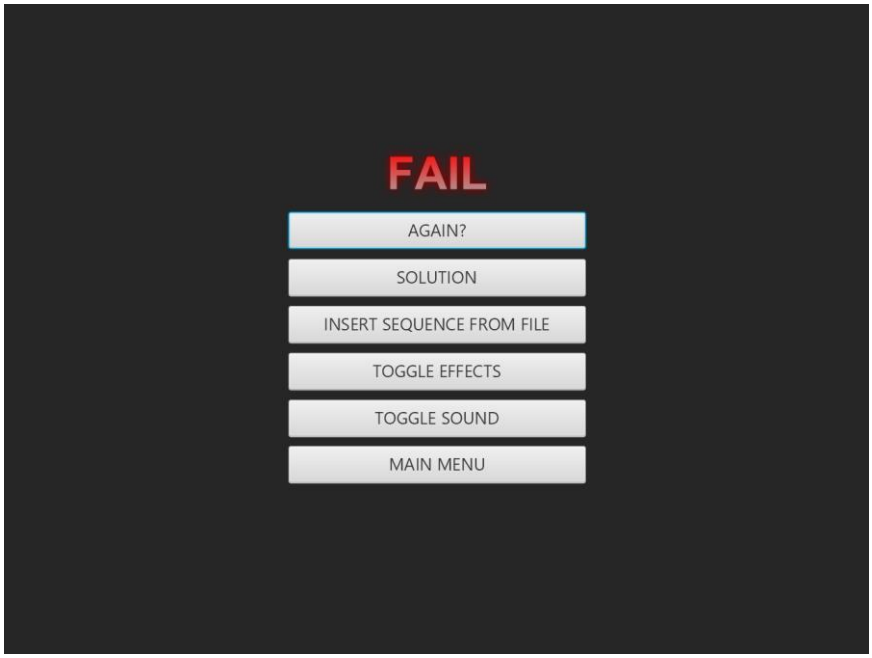



***Слика 5.2.1 – Играчки интерфејс тренинг нивоа***

### *5.2.2. Мени паузе – тренинг мод*

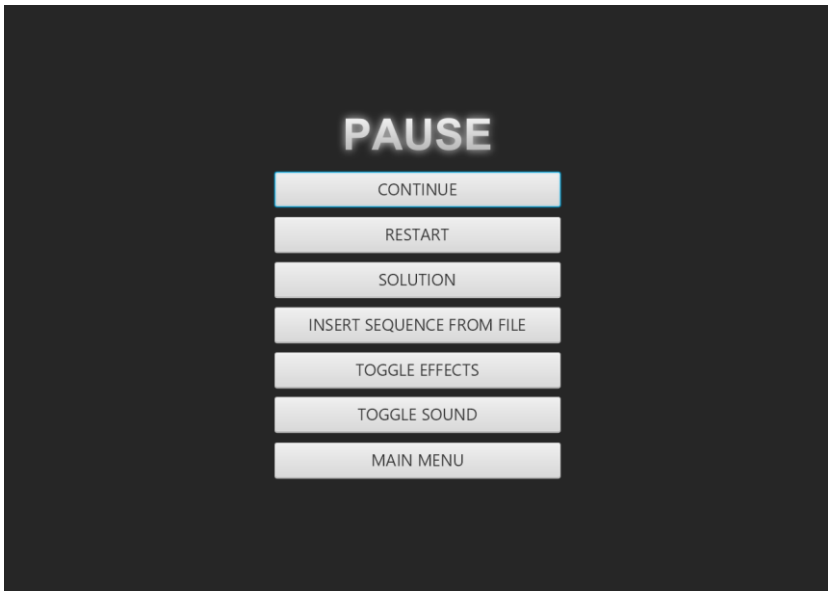
Уколико корисник успешно заврши ниво или пак изгуби, приказује му се мени са следећим опцијама (Слика 5.2.2а):

- AGAIN? – поновно покретање истог нивоа
- SOLUTION – опција која решава ниво и решење записује у текстуални фајл
- INSERT SEQUENCE FROM FILE – омогућава одигравање секвенце потеза њеним учитавањем из текстуалног фајла
- TOGGLE EFFECTS – гашење/паљење звучних ефеката
- TOGGLE SOUND - гашење/паљење позадинске музике
- MAIN MENU – повратак на почетни мени




***Слика 5.2.2а – Мени паузе – крај тренинг нивоа***

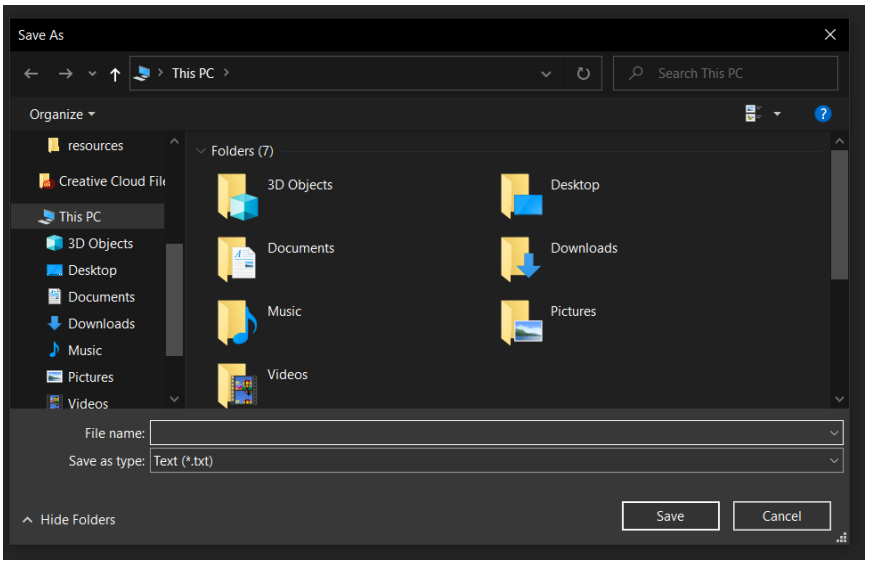
У горњем десном углу играчког интерфејса налази се дугме за одабир паузе. Кликом левог тастера миша на ово дугме или притиском тастера ESC са тастатуре отвара се мени паузе који уместо AGAIN опције садржи CONTINUE опцију којом играч наставља игру из позиције у којој је она заустављена и RESTART којом се блок враћа на почетну позицију а број одиграних потеза враћа на 0 (Слика 5.2.2б).




***Слика 5.2.2б – Мени паузе након клика на PAUSE опцију***

### *5.2.3. Решење*

Кликом на дугме SOLUTION отвара се нови Save As прозор путем ког корисник одабира где ће сачувати решење нивоа (Слика 5.2.3а). Решење представља један од могућих низа секвенци којима се може решити ниво записан у виду редоследа потеза које корисник треба да одигра да би победио. На слици 5.2.3б дат је изглед решења једног од тренинг нивоа.




***Слика 5.2.3а – Save As мени***

```
1  up
2  up
3  right
4  up
5  up
6  up
7  up
8  right
```

***Слика 5.2.3б – Изглед решења првог тренинг нивоа***

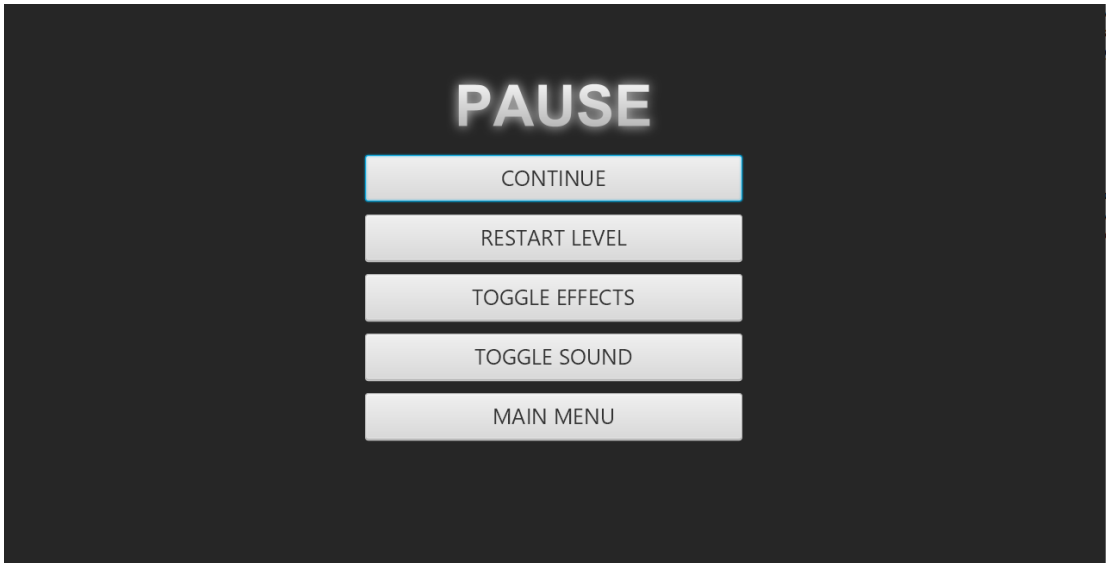
### *5.2.4. Старт*

Одабиром опције START из почетног менија корисник започиње нову игру која се бодује. Треба имати у виду да се ова игра састоји из низа нивоа који су поређани по тежини од лакшег ка тежем и да је за прелазак целе игрице потребно прећи све нивое. У горњем левом углу екрана корисник може да прати број одиграних потеза од тренутка стартовања игре. Уколико корисник изгуби тренутни ниво, блок се враћа на почетну позицију а број потеза се не мења – сабира се сваки потез.

### *5.2.5. Мени паузе – такмичарски мод*

Мени паузе се односи на опцију у игри која кориснику омогућава да привремено заустави игру. У току игре корисник може одабрати опцију PAUSE која се налази у горњем десном углу. Овај мени значајно се разликује од менија који се отвара приликом завршетка тренинг нивоа јер садржи следеће опције (Слика 5.2.5):

- CONTINUE – наставити игру
- RESTART LEVEL – враћа блок на почетну позицију, а број одиграних потеза остаје исти
- TOGGLE EFFECTS – гашење/паљење звучних ефеката
- TOGGLE SOUND – гашење/паљење позадинске музике
- MAIN MENU – повратак на почетни мени




***Слика 5.2.5 – Мени паузе – такмичарки мод***

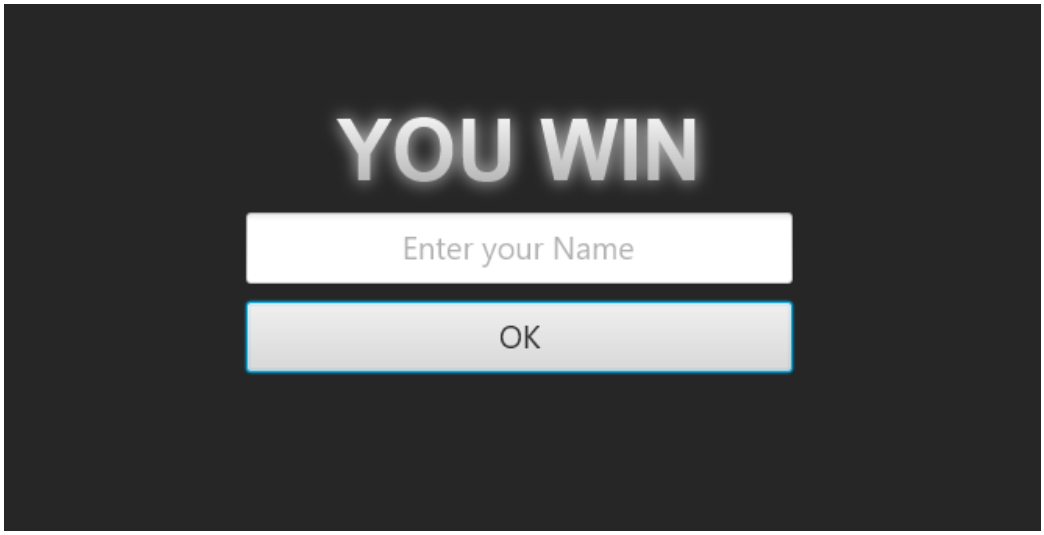
Као што се може приметити овај мени не садржи опције за решавање нивоа и учитавање предефинисане секвенце потеза да би игра била захтевнија и занимљивија.

Тастери за управљање звучним ефектима и музиком додати су да играч не би морао да прекида игру уколико у неком тренутку пожели да упали/угаси исте.

Мени паузе се такође може отворити кликом на ESC дугме на тастатури.

### *5.2.6. Победнички мени*

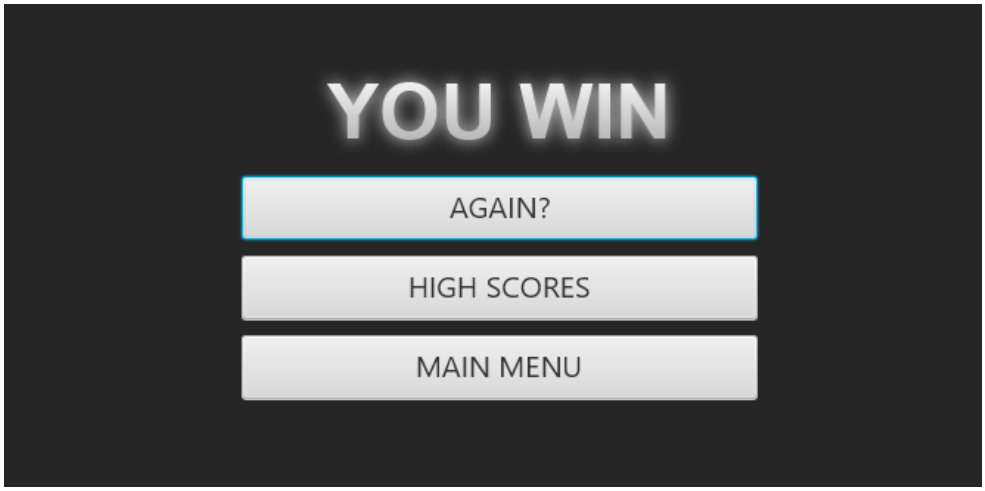
Након што корисник пређе све нивое, игрица се завршава и отвара се мени где може уписати своје име у одређено поље и кликом на OK наставити даље (Слика 5.2.6а). Уколико је корисник освојио довољан број бодова у току игре и остварио услов да се упише у листу најбољих резултата, име које је унео у овом менију биће уписано заједно са бројем бодова у листу најбољих резултата. Уколико је играч заборавио да унесе име и прескочио овај корак, његов резултат биће анониман (уместо имена писаће *Anonymous*).




***Слика 5.2.6а – Победнички мени***

Кликом на OK отвара се нови мени који омогућава следеће опције (Слика 5.2.6б):

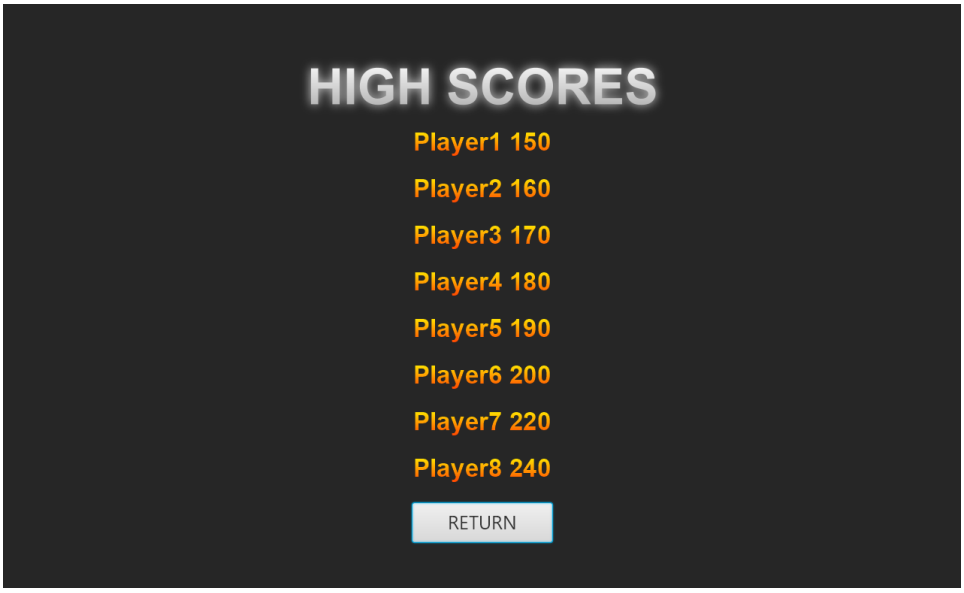
- AGAIN? – поновно покретање игрице од првог нивоа
- HIGH SCORES – листа најбољих резултата
- MAIN MENU – повратак на почетни мени




***Слика 5.2.6б – Додатне опције победничког менија***

### *5.2.7. Листа најбољих резултата*

Одабиром опције HIGH SCORES отвара се ранг листа 8 најбољих играча. Уколико корисник први пут игра игру или ако је обрисао конфигурациони фајл, листа ће бити попуњена аутоматски генерисаним именима анонимних играча и њиховим измишљеним резултатима (Слика 5.2.7). Уколико је корисник успешно завршио све нивое и постигао довољно добар резултат његово име биће уписано у дату листу.

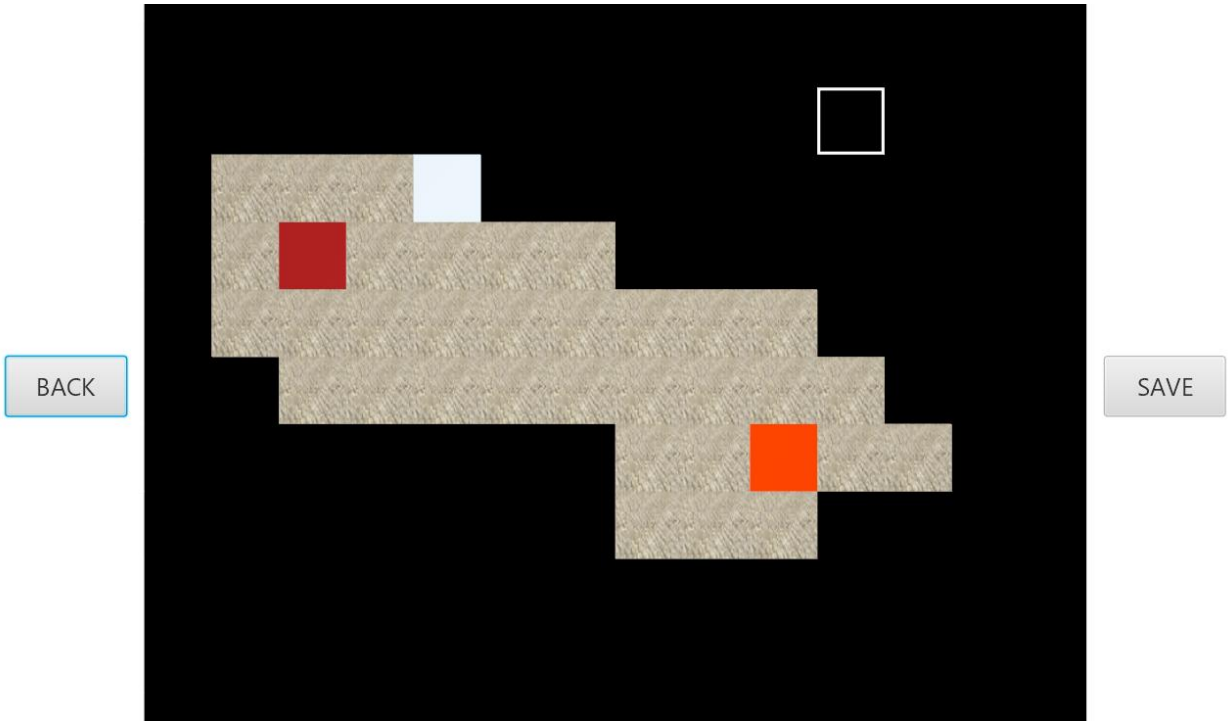



***Слика 5.2.7 – Иницијална листа најбољих резултата***

Притиском на дугме RETURN корисник се враћа на почетни мени.

## 5.3. Креирање нивоа

На почетном менију постоји опција CREATE LEVEL која кориснику омогућава слободу прављења сопствених нивоа (Слика 5.3). Постоје 4 основна шаблона које корисник може да одабере.

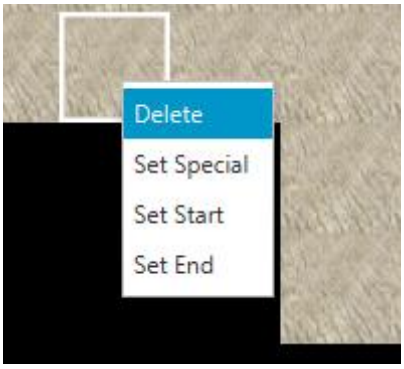



***Слика 5.3 – Изглед корисничког интерфејса приликом креирања нивоа***

### *5.3.1. Подешавање обичног поља*

Приликом креирања нових нивоа, постоје различита правила по којима се полигон може направити и то у зависности од поља које се подешава, па тако левим кликом показивача на обично поље појављују се следеће опције (Слика 5.3.1а):

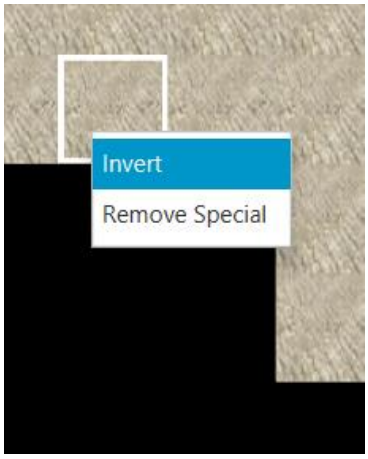
- DELETE – брисање постојећег поља – појављује се само уколико је поље на ивици полигона
- SET SPECIAL – постављање специјалног поља
- SET START – померање старта/места са ког блок полази на тренутно поље
- SET END – померање циља до ког блок треба да стигне на тренутно поље




***Слика 5.3.1а – Леви клик показивачем на обично поље***

Десним кликом показивача појављују се опције које су једнаке за сва поља (Слика 5.3.1б):

- INVERT – кликом на ову опцију старт, односно место са ког блок полази и циљно место мењају своје позиције
- REMOVE SPECIAL – уклањање свих специјалних поља са полигона при чему се она мењају обичним пољима

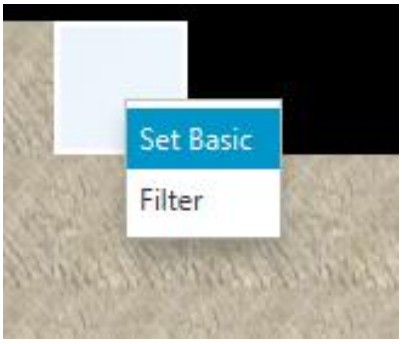



***Слика 5.3.1б – Десни клик показивачем на обично поље***

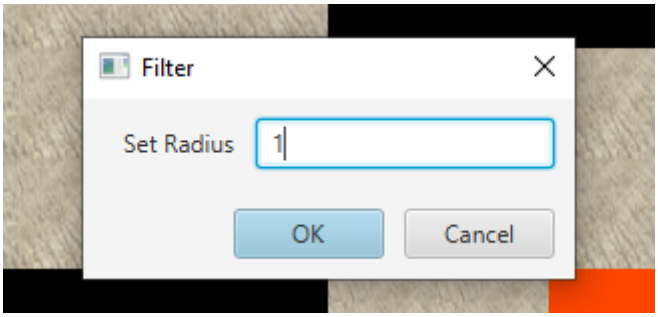
### *5.3.2. Подешавање специјалног поља*

Левим кликом показивача на специјално поље приказују се следеће опције (5.3.2а):

- SET BASIC – враћање специјалног поља у обично
- FILTER – кликом на опцију филтер остварује се могућност брисања специјалних поља у радијусу који корисник накнадно уноси. Уколико је радијус 0 само одабрано специјално поље биће конвертовано у обично (Слика 5.3.2б).



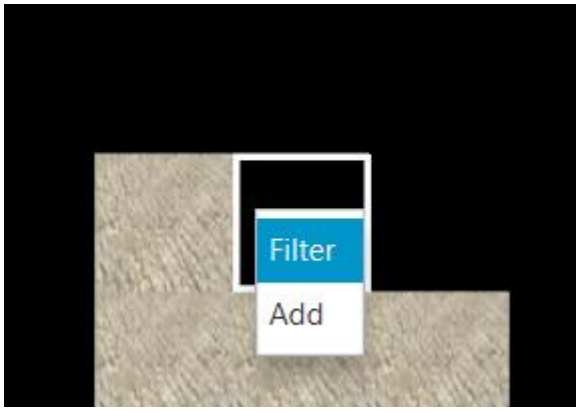

*Слика 5.3.2а – Опције специфичне за специјално поље*




*Слика 5.3.2б – Филтер опција*

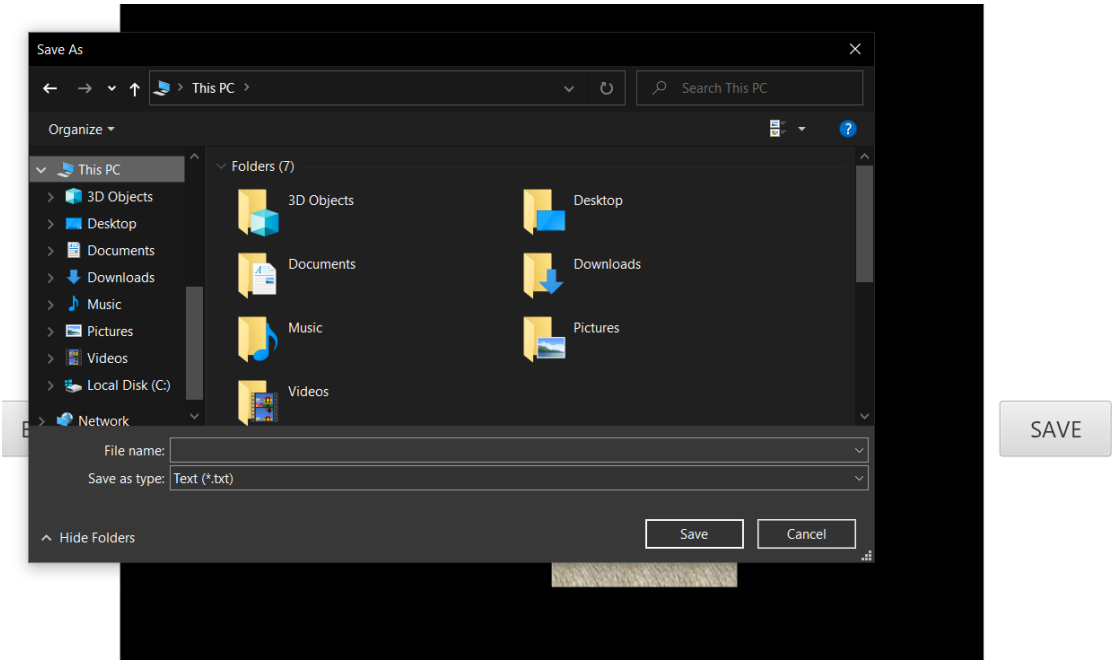
### *5.3.3. Подешавањепразног поља*

Полигон се може додатно проширити конвертовањем празних поља у обична и њиховим даљим подешавањем. У овом случају постоји правило да само празна поља која се налазе уз полигон могу бити подешена (Слика 5.3.3а). Уз ову конверзију, опцији за брисање специјалних поља у дефинисаном радијусу могуће је приступити левим кликом на празно поље.




*Слика 5.3.3а – Додавање новог поља*

Додатно дугме EXIT које се налазе са леве стране едитора служи за напуштање режима за креирање нивоа и повратак на почетни мени, док SAVE дугме које се налази са десне стране отвара прозор за одабир локације на којој ће нови ниво бити сачуван (Слика 5.3.3б).




***Слика 5.3.3б – Одабир локације за чување нивоа након клика на SAVE дугме***

# 6. Закључак

Инспирација за развој овакве апликације произашла је из жеље да се савлада рад са *ScalaFX* библиотеком и концептима 3Д рачунарске графике, као и да се стекне разумевање изазова и проблема који се јављају у току креирања видео игара. У оквиру пројекта су дизајниране и успешно интегрисане функционалности за цртање 3Д облика. Додатно, развијена је игра која подстиче такмичарски дух играча, тестира његову логику и сналажљивост и пружа могућност за креативност и неограничено прављење нових нивоа игре. Те особине могле би се искористити у сврхе организовања неких мањих надметања, те додељивања награде најбољем играчу.

Игра поседује могућност прављења великог броја различитих полигона, међутим могао би се увести додатан одабир и модификација димензије терена. Затим додатни типови поља са још различитих функционалности попут поља на које блок не би смео да стане хоризонтално или блокова који би представљали дугмад након чијег притиска би се појављивали делови терена који би фалили.

Још пар ствари које би се могле додати у будућности и које би унапредиле игру јесу:

- увођење неограниченог броја нивоа са могућностима брисања и подешавања сваког посебно, односно посебна апликација – едитор која би служила за лакше прављење нових верзија апликација. Апликација би могла да садржи и алгоритам за аутоматско генерисање мапа
- увођење више тежина нивоа попут супер лаког, лаког, средњег и тешког
- унапређење визуелне атрактивности игре и корисничког интерфејса по узору на различите 3Д видео игре из жанра слагалица
- опција за одабир теме игре
- увођење ограниченог времена за решавање нивоа у сврху напреднијег тестирања сналажљивости и концентрације играча.

Такође, било би интересантно, по узору на *Bloxorz 2*, направити игру инспирисану неким посебним мотивом и пољима која би имала карактеристичне акције у складу са одабраном темом.

Иако на први поглед ова игрица може изгледати као игра с једноставним концептом, унутар ње крије се цео свет неограничених могућности. Сваки корак играча отвара врата ка новим искуствима и стратегијама које се откривају како се игра развија. Ова игрица једноставно доказује да велика количина забаве може доћи из основних концепата уколико се они довољно добро промисле и адекватно примене. Такође, игрица пружа многобројне могућности за креирање нових врста поља, препрека и нивоа.

# Литература


[1] Marçal Mora-Cantallops, Transhistorical perspective of the puzzle video game genre, August 2018, Conference: the 13th International Conference

https://www.researchgate.net/publication/327639714_Transhistorical_perspective_of_the_puzzle_video_game_genre (посећен дана 06.09.2023.)

[2] Историја игара-слагалица, *GameSpot* страница,

https://web.archive.org/web/20100204081152/http://www.gamespot.com/features/vgs/universal/puzzle_hs/ (посећен дана 05.09.2023.)

[3] *Puzzle video game*, Википедија страница,

https://en.wikipedia.org/wiki/Puzzle_video_game (посећен дана 14.09.2023.)

[4] Информације о Дамијан Кларку, Дамијан Кларк званична страница,

https://damienclarke.me/ (посећен дана 06.09.2023.)

[5] Миниклип платформа, Општи преглед

https://www.miniclip.com/our-story (посећен дана 15.09.2023.)

[6] Cher Wee Ang, Cognitive training with casual video games: The effects of working memory and reasoning related games on the cognitive abilities of younger and older adults, PhD Dissertation, University of Illinois at Urbana-Champaign, 2016

[7] Pauline L. Baniqued, Michael B. Kranz, Michelle W. Voss, Hyunkyu Lee, Joshua D. Cosman, Joan Severson and Arthur F. Kramer, Cognitive training with casual video games: points to consider, part of the Video Games as Tools to Achieve Insight into Cognitive Processes, Frontiers in Psychology [online]:

https://www.frontiersin.org/research-topics/1427/video-games-as-tools-to-achieve-insight-into-cognitive-processes#articles (посећен дана 13.09.2023.)

[8] Peter Blanchfield, Kamal Basha Madarsha, Mohd. Syarqawy Hamzah, The use of computer games in training spatial reasoning, [online]:

https://www.scitepress.org/Papers/2012/38839/38839.pdf (посећен дана 06.09.2023.)

[9] Adobe Flash званична страница, Општи преглед

https://get.adobe.com/flashplayer/about (посећен дана 06.09.2023.)

[10] Definition of ActionScript, Software Engineer Insider, (посећен дана 14.09.2023.)

https://www.softwareengineerinsider.com/programming-languages/actionscript.html

[11] Објашњење и дефиниција игре *Cuboid*, Википедија страница,

https://en.wikipedia.org/wiki/Cuboid_(video_game) (посећен дана 06.09.2023.)

[12] Metacritic страница, Општи преглед

https://www.metacritic.com/about-us/ (посећен дана 15.09.2023.)

[13] Функционално програмирање - парадигма, geeksforgeeks.org,

https://www.geeksforgeeks.org/functional-programming-paradigm/ (посећен дана 06.09.2023.)

[14] *Ламбда рачун,* Википедија страница,

https://en.wikipedia.org/wiki/Lambda_calculus (посећен дана 06.09.2023.)

[15] *Scala* програмски језик званична страница, Општи преглед,

https://www.scala-lang.org/ (посећен дана 06.09.2023.)

[16] *ScalaFX* званична страница: Општи преглед,

https://www.scalafx.org/ (посећен дана 06.09.2023.)

[17] *JavaFX* званична страница: Општи преглед,

https://openjfx.io/ (посећен дана 06.09.2023.)

[18] Тартаља И., Материјали за предавања из предмета Рачунарска графика, 2020.,

https://rti.etf.bg.ac.rs/rti/ri5rg/materijali/predavanja/12%20JavaFX_8.%20deo%20-%20kamera.pdf (посећен дана 13.09.2023.)

[19] *IntelliJ IDEA* развојно окружење, званична веб страница:

https://www.jetbrains.com/idea (посећен дана 06.09.2023.)

[20] Репна рекурзија, geeksforgeeks.org:

https://www.geeksforgeeks.org/tail-recursion-in-scala/ (посећен дана 06.09.2023.)

[21] Основни закони динамике ротације, момент силе, Физика за први разред гимназије, [online]:

https://fizis.rs/гимназија/i-разред/динамика-ротационог-кретања-крутог-т/osnovni-zakon-dinamike-rotacije/ (посећен дана 14.09.2023.)

[22] Равнотежа тела, физика за 7. Разред,

http://fizis.rs/osnovna-skola/vii-razred/равнотежа-тела/pojam-i-vrste-ravnoteze-tela/ (посећен дана 14.09.2023.)

# Списак слика

# СПИСАК ТАБЕЛА