Универзитет у Београду - Електротехнички факултет   
Катедра за рачунарску технику и информатику



##### Развој лабораторијског задатка из 3D графике – видео игра преживљавања у Арени

дипломски рад основних академских студија

|  |  |
| --- | --- |
| Ментор | Студент |
| др Игор Тартаља, в. проф. | Војислав Богосављевић  94/2014 |
|  |  |

Београд, 2020.

# Апстракт

У овом раду примарно је описано искуство развоја 3D видео игре и њена еволуција од скелета кода показне лабораторијске вежбе, преко контролне лабораторијске вежбе и домаћег задатка (пројекта), све до нивоа дипломског рада. Софтвер је израђен на програмском језику *Java*, уз коришћење бибилиотеке *JavaFX*.

Тема лабораторијских вежби је 3D игра *Arena*, у којој је играчу циљ да за ограничено време сакупи што већи број поена и преживи у арени избегавањем препрека. Вежбе служе за проверу стеченог знања студената: полазећи од приложеног скелета кода, задатак је да имплементирају додатне функционалности на контролној лабораторијскох вежби. То се даље може надоградити кроз домаћи задатак, те још више кроз дипломски рад; у оба случаја се захтеви додатно усложњавају и повећава им се обим.

И поред тога што је *JavaFX* предвиђена да буде довољно једноставна за употребу и да уз не много напора омогући постизање релативно једноставних (визуелних) резултата, развој игре се по свим нивоима показао као пријатно изазован. Све време су били присутни проблеми писања добро структурираног и лако проширивог кода, осмишљавање алгоритама и њихова интеграција са оним што пружа библиотека, као и један делом уметнички аспект – дизајн игре и искуства које пружа.

У рад је укључен и низ прилога, попут упутстава за коришћење и одржавање програма, текстови лабораторијских вежби и домаћег задатка. Такође је развијено и више артефаката као што су изворни кодови по различитим нивоима развоја и презентација за показну лабораторијску вежбу.

Садржај

[Апстракт 2](#_Toc36595916)

[1. Увод 4](#_Toc36595917)

[2. Проблем 5](#_Toc36595918)

[3. Преглед сродних игара 6](#_Toc36595919)

[3.1 *Rayman 2: The Great Escape* 7](#_Toc36595920)

[3.2 *Croc 2* 8](#_Toc36595921)

[3.3 *Half-Life* 9](#_Toc36595922)

[4. Спецификација лабораторијских вежби и домаћег задатка 10](#_Toc36595923)

[5. Функционална спецификација игре 12](#_Toc36595924)

[6. Пројекат игре 14](#_Toc36595925)

[7. Имплементација игре 16](#_Toc36595926)

[7.1 Техничке карактеристике 16](#_Toc36595927)

[7.2 Занимљиви детаљи имплементације 17](#_Toc36595928)

[7.2.1 Како је пројекат започет 17](#_Toc36595929)

[7.2.2 Перспектива из првог лица 17](#_Toc36595930)

[7.2.3 *Stream* 18](#_Toc36595931)

[8. Закључак 20](#_Toc36595932)

[9. Литература 21](#_Toc36595933)

# 1. Увод

Рачунарска графика је наука и уметност комуникације визуелним путем помоћу рачунарског дисплеја и уређаја за интеракцију [1]. Њен циљ је синтеза статичке или покретне слике уз интеракцију са корисником. Рачунарска графика прожима скоро све области људске делатности, а извесно је да ће, како човечанство даље буде напредовало у технолошком смислу, ослањање на рачунарску графику бити још веће. Поред очигледних значајних области примене, попут науке, медицине, образовања и слично, постоји и једна у којој је графика од суштинске важности – штавише, могло би се тврдити и да веома репрезентативно осликава успехе које људска врста постиже по питању графике – а то је индустрија игара.

На Електротехничном факултету Универзитета у Београду један од предмета који нуди изучавање поља о ком је реч јесте *Рачунарска графика*. У оквиру тог предмета се изучавају основни концепти и добија теоријска подлога за даље продубљивање знања, а такође се стечена знања примењују и утемељују кроз практичне делове предмета. Тај практични део, који ће бити детаљније описан у наредним одељцима, је изнедрио и овај рад.

На поменутом курсу се за реализацију практичних задатака користи програмски језик *Java* и библиотека за њу, *JavaFX*, која омогућава довољно једноставно и ефикасно постизање резултата у 2D и 3D графици [2]. Због своје једноставности и пријемчивости (а без губљења свеобухватности), добар је избор за академске сврхе. Развојно оружење у ком се ради је *NetBeans*, који нуди подршку за *JavaFX* [3].

У овом раду описан је еволутивни развој једне 3Д видео-игре као основе за израду најпре лабораторијских вежби, затим домаћег задатка (пројекта) и, на крају, дипломског рада. Тема игре је *Arena* са три просторије у којима играч настоји да преживи одређено време у борби са различитим врстама препрека у различитим просторијама, како што су шиљци који искачу из тла, пројектили који прелећу кроз просторију, односно сечива која падају са плафона.

У следећем поглављу се описује решавани проблем, у трећем је дат преглед игара које су сродне *Areni* или су послужиле као инспирација за њу, у четвртом су описи лабораторијских вежби и домаћег задатка. У петом поглављу је сажета функционална спецификација игре (у прилогу А је дато упутство за коришћење са детаљном функционалном спецификацијом програма); у шестом кратак преглед најважнијих делова архитектуре програма (у прилогу Б је дато упутство за одржавње са детаљним моделом програма). У седмом поглављу је описана имплементација игре, приказане су техничке карактеристике, као и неколико детаља имплементације који читаоцу могу бити занимљиви. У осмом поглављу је наведен закључак рада, а у деветом је излистана литература коришћена при писању овог рада.

# 2. Проблем

Знања усвојена на курсу Рачунарска графика студенти имају прилику да практично примене кроз лабораторијске вежбе. Организују се четири вежбе, од којих су прва и трећа показног карактера, а друга и четврта контролног; прве две везане за 2D графику, друге две за 3D.

На контролним лабораторијским вежбама студенти се упознају са програмом (скелетом кода) који ће на контролној вежби надограђивати за поене. Скелет кода садржи основне функционалности и представља смислену целину. Студенти тај скелет обогаћују додатним (мањим) функционалностима и кроз то утврђују своја знања, показују у којој су мери савладали материју и вежбају да врше измене, односно проширења над туђим кодом.

Као тема треће и четврте лабораторијске вежбе одабрана је 3Д видео-игра. Мотивација за то је чињеница да проблем програмирања видео-игре обухвата више различитих аспеката: најпре, релевантан за овај рад, визуелни дизајн и реализација графичког дела, осмишљавање самог искуства које игра пружа, алгоритми за спровођење тих замисли у дело, писање квалитетно структурираног, одрживог и проширивог кода, те дизајн звука (позадинске музике и звучних ефеката) и друго [4]. Такође је значајно што се исходи развоја јако брзо виде, односно потребно је релативно мало времена и труда да би се постигли видљиви резултати, што може бити веома мотивишуће студентима.

Практични део курса Рачунарска графика се уоквирује кроз домаћи задатак. Међу понуђеним задацима постоји опција даље надоградње лабораторијске вежбе (над реализованим свим ставкама за контролну лабораторијску вежбу) и захтеви за то су осетно сложенији и обимнији. Напослетку, полазећи од нивоа домаћег задатка, игра се може додатно проширити још комплекснијим функционалностима, како би нарасла на ниво дипломског или мастер рада.

# 3. Преглед сродних игара

*Arena* је жанровски најближа општој категорији акционих игара (енг. *action game*) у којима је нагласак на физичким изазовима, што укључује координацију између рукe и ока и брзину реакције [5]. Могла би се повући и паралела са платформским играма (енг. *platformer*) као поджанром акционих, у којима је циљ, у општем случају, скакати и пењати се до одвојених платформи уз избегавање препрека [6]. Међутим, чест је случај да платформске игре буду из перспективе трећег лица, што овде није случај. Перспектива у игри *Arena* је (примарно) из првог лица (енг. *first-person*), рађена најпре по угледу на пуцачине из првог лица [7]. Такође је одсутна могућност скакања пошто се све одиграва у једној хоризонталној равни.

*Arena* није рађена као клон ниједне игре, али поједини аспекти јесу реализовани по узору на друге. Игре које из којих је аутор црпео инспирацију дате су у наставку.

## 3.1 *Rayman 2: The Great Escape*

*Rayman 2: The Great Escape* [8] (Слика 3-1) је платформска видео-игра у којој играч контролише Рејмена, хуманоидног лика карактеристичног по одсуству удова, чије шаке, стопала и глава лебде одвојени од торзоа. Он има могућност гађања противника енергетским пројектилима, а додатну контролу кретања приликом скокова му дозвољава покретање косе у маниру елисе хеликоптера. Радња игре се врти око инвазије робота-гусара из свемира који намеравају да окупирају свет. Игру карактеришу квалитетно дизајнирани нивои, свеприсутна мистична атмосфера и упечатљиви ликови. Елементи игре који су послужили као инспирација за *Arenu* јесу разноврсне препреке, попут пипака који искачу из зидова и пројектила којима противници гађају играча.



Слика 3-1

## 3.2 *Croc 2*

*Croc 2* [9] (Слика 3-2) је платформска видео-игра у којој играч контролише Крока, крокодила у потрази за несталим родитељима. Играч има разне могућности на располагању што се тиче контроле, као што су вишеструки скок и различите врсте напада. Свет је доста отворен и често је на играчу да одабере којим ће редоследом прелазити поставе. За ову игру је карактеристичан велики број разноврсних нивоа и њихов концептуални и визуелни дизајн, од којих су неки елементи послужили као узор за *Arenu*, међу којима су препреке попут искачућих шиљака и срца за сакупљање ради обнављања изгубљених живота.



Слика 3-2

## 3.3 *Half-Life*

У питању је једна од најпознатијих и најутицајнијих игара жанра пуцачине из првог лица, која се убраја међу најбоље игре свих времена [10]. Срж сценарија игре (енг. *gameplay*) је борба против ванземаљских и људских противника и решавање загонетки (Слика 3-3). Играч контролише Гордона Фримена, научника који се бори за свој живот након што у истраживачком центру експеримент са ванземаљским материјалом пође наопако. Иако *Arena* жанровски није пуцачина, генерални осећај контроле из првог лица и елементи визуелног идентитета (на пример, величина видног поља, брзина кретања и дизајн шиљака и сечива) црпели су инспирацију из игре *Half-Life*; ваља нагласити и да ова игра и те како поседује платформске елементе.



Слика 3-3

# 4. Спецификација лабораторијских вежби и домаћег задатка

На показној лабораторијској вежби студенти најпре пролазе кроз кратке демонстрационе задатке у којима им се објашњавају концепти програмирања 3D графике. Потом се изводи главни део вежбе – упознавање са скелетом кода намењеног за проширивање (и описаног у Прилогу Ц), уз објашњења демонстратора и пропраћено презентацијом (присутна међу приложеним артефактима). После презентовања, последњи део посвећен је самосталној вежби, док је демонстратор ту да одговара на евентуална питања и асистира у решавању. Текст показне лабораторијске вежбе може се наћи у Прилогу Д1, док је упутство за спровођење прве лабораторијске вежбе дато у Прилогу Д2 .

Скелет кода у овом конкретном случају садржи основне функционалности погодне за даљу надоградњу: просторију у арени, руковаоце догађајима за корисничку контролу, детекцију колизија, тајмер за покретање дешавања у игри и друго. Пројекат и извршни код скелета и решења показне лабораторијске вежбе приложени су као софтверски артефакти.

Контролна лабораторијска вежба служи за проверу знања студената. Ту је потребно да се за ограничено време (2 сата) прошири скелет кода. Све понуђене функционалности које студент треба да уведе на контролној лабораторијској вежби нуде више од максималних 100 поена, те је на студенту да одабере које му одговарају како би постигао што више за расположиво време. Такође, на увид му је дато и решење лабораторијске вежбе у извршном облику, тако да може да види чему његов рад оквирно треба да тежи приликом имплементације. Основне функционалности које студент треба да реализује на контролној лабораторијској вежби су: креирање новчића и интеракције између играча и новчића, креирање цветова као објеката са којима играч не интерагује, додавање исписа преосталог времена и порука на екрану, додељивање текстура појединим моделима, промена модела шиљака у пирамидални, додавање тачкастих извора светлости, као и додавање камере која може да се ротира и помера и омогућавање промене погледа између првог лица и погледа те камере. Комплетна листа задатака дата је у тексту контролне лабораторијске вежбе која се налази у прилогу Е1, упутства за спровођење ове вежбе у Прилогу Е2, а пројекат и извршни код решења вежбе су међу приложеним софтверским артефактима.

Домаћи задатак (пројекат) се ради самостално и брани у испитним роковима по завршетку семестра. У питању су сложенији програмерски проблеми и студент има избор између неколико понуђених задатака. Опција међу њима је и увођење додатних функционалности над комплетираним захтевима контролне лабораторијске вежбе. Најважнији додатни захтеви за домаћи задатак су: креирање нових предмета које играч сакупља, као што су сатови који продужавају расположиво време и срца која обнављају животе, затим онемогућавање да се на истој плочи налази више предмета, као и увођење могућности брзог трчања. Домаћи задатак има могућност даљег развоја до нивоа дипломског и мастер рада; као резултат тога, настао је и овај рад, као надоградња апликације *Arena* до нивоа комплетиране релативно једноставне 3Д видео-игре. Домаћи задатак који представља надоградњу видео игре *Arena* описан је у Прилогу Ф1, а упутствo за одбрану домаћег задатка у прилогу Ф2. Пројекат и извршни код домаћег задатка су међу приложеним софтверским артефактима.

# 5. Функционална спецификација игре

У овом поглављу дат је кратак преглед главних функционалности реализованих у оквиру овог рада. Kомплетно упутство за коришћење са детаљним описом функционалности игре дато је у прилогу А.

*Arena* је 3Д акциона игра из првог лица и играчев циљ је да сакупи што више поена за ограничено време које му је на располагању, уз избегавање препрека. Постоје 3 просторије у Арени, свака са својом врстом препрека.

По покретању игре, корисника дочекује главни мени са опцијама да покрене игру, види табелу најбољих резултата, уђе у подешавања и напусти програм. Пре започињања игре неопходно је унети име играча.

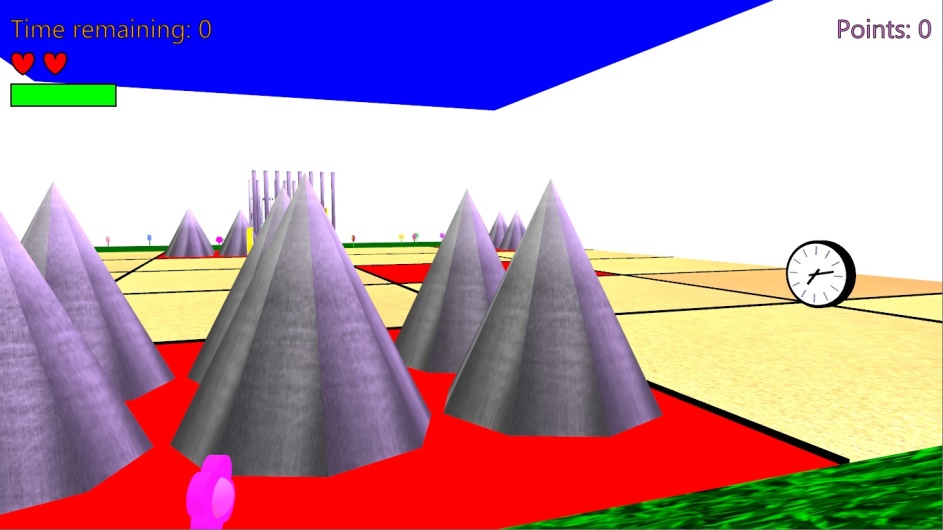
Основне команде за управљање играчем су: тастери *W*, *S*, *A* и *D* за кретање у различитим смеровима, померањем миша се усмерава камера из првог лица, односно мења оријентација играча, држањем тастера *SHIFT* се брзо трчи, а на тастер *ESC* се игра паузира, односно наставља. Кад је игра паузирана, приказује се посебан мени са опцијама да се игра настави, да се покрене из почетка, да се врати у главни мени и да се изађе из програма.

Арена је подељена на 3 просторије, свака са различитим врстама препрека: шиљци (Слика 5-1), пројектили (Слика 5-2) и сечива (Слика 5-3). При контакту са њима, играч губи један живот. Над пољима на које је тло просторија подељено стварају се предмети за сакупљање: новчићи који дају поене, срца која обнављају животе и сатови који дају додатно време (на сликама просторија могу се видети и ти предмети).

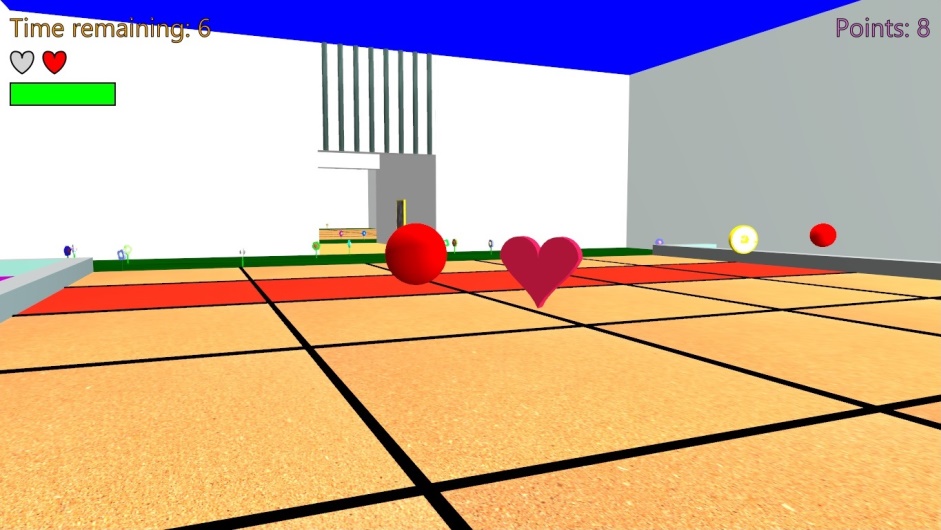
Играчев интерфејс (енг. *Heads-up Display*, HUD), стално приказан током игре, садржи информације о преосталом времену, животима и енергији за брзо трчање (такође се могу видети на сликама просторија). Енергија се троши док играч брзо трчи, а у супротном обнавља.

Одбројавање времена у просторији почиње чим играч закорачи у просторију из ходника који јој претходи. Након истека времена у једној просторији, отварају се капије ходника ка наредној.

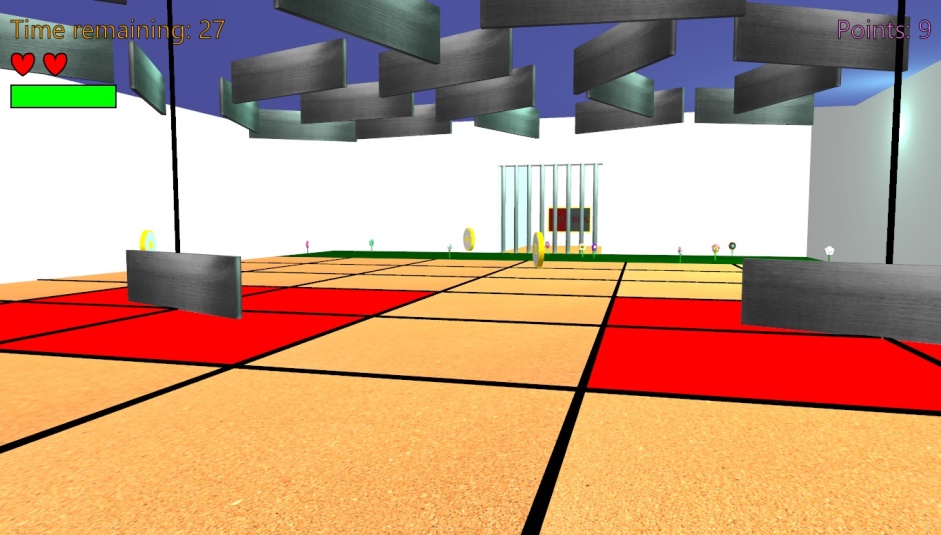
Игра се успешно завршава уколико играч дочека истек времена у последњој просторији, а неуспешно ако изгуби све животе.



Слика 5-1



Слика 5-2

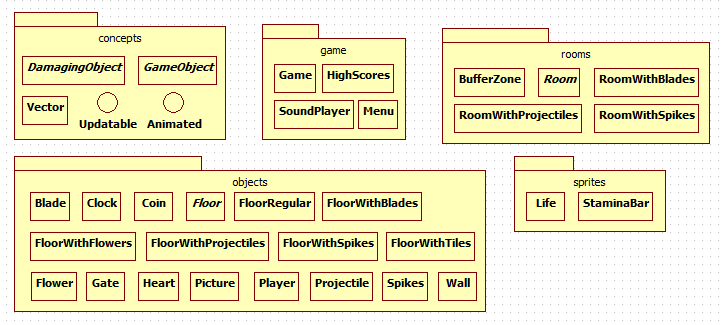


Слика 5-3

# 6. Пројекат игре

У овом поглављу дат је кратак преглед најважнијих делова архитектуре игре. Целовит UML модел игре може се наћи у прилогу Б.

На Слици 6-1 приказан је дијаграм пакета са свим класама и интерфејсима у пројекту.



Слика 6-1

У пакету Concepts налазе се класе и интерфејси који су основа за велики број осталих класа: GameObject је наткласа свих класа у пакетима objects и rooms, Vector је присутан као поље у класи GameObject и као параметар њеног конструктора, док интерфејсе Updatable и Animated имплементирају многе класе из наведених пакета, као и пакета sprites.

Класе које представљају општије целине и покретаче дешавања у игри налазе се у пакету game. Међу њима је главна класа Game у којој је и main() метода. Та класа је кључна за ток игре, омогућава паузирање и прелазак у мени, а такође поседује тајмер на чије се откуцаје одигравају многе важне ствари, попут изазивања догађаја у одређеним тренуцима, детекције колизија, рачунања протока времена и слично. Практично повезује све класе у пројекту.

Пакет objects садржи 3Д објекте који се појављују у игри, као што су сатови, тло, зидови и слично. Ту је и класа Player која има нешто сложеније могућности: представља играча, има носач за камеру која служи као његов поглед, руковаоце догађајима за кориснички улаз ради контроле лика и врши значајан део управљања и кретања по арени.

Иако представљају 3Д објекте, односно просторије у арени, класе унутар пакета rooms су у њега издвојене из разлога што су у питању објекти сложени од већег броја оних који представљају засебне целине, из пакета objects. Такође, поседују функционалности које утичу на дешавања у игри и нису само статичне целине које служе да буду коришћене од стране других делова кода (опет, као већина класа у пакету objects).

У sprites су смештени 2Д објекти који су део корисничког интерфејса – срца која представљају преостале животе и показивач преостале енергије за брзо трчање.

# 7. Имплементација игре

Основа за имплементацију игре је скелет кода. Полазећи од њега, игра се проширује до нивоа контролне лабораторијске вежбе са комплетираним свим захтевима. Те измене су прилагођене ограниченом времену израде лабораторијске вежбе, те су махом естетске природе, са изузетком додавања новчића, чиме се игри назире поента. Одатле, реализацијом ставки за домаћи задатак, апликација почиње да поприма опште црте познате за велики број игара: ограничено време за сакупљање што већег броја поена и могућност губитка живота чине игру изазовном и отварају могућност постизања упоредивих резултата по одиграним партијама, док постојање обновљивих залиха енергије за брзо трчање и појава сатова за додатно време наводи играча да се паметно креће по арени. Надоградња од нивоа домаћег задатка до дипломског рада комплетира видео-игру: више просторија даје већу слободу кретања и представља мали свет за себе, позадинска музика и ефекти доприносе атмосфери, а постојање менија и могућност паузирања игре наговештавају да је у питању целовито решење, налик модерним играма.

## 7.1 Техничке карактеристике

У Табели 7-1 приказане су техничке карактеристике решења игре по различитим верзијама. За анализу је коришћен програм *SourceMonitor* [11].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Скелет кода | Контролна лабораторијска вежба | Домаћи задатак | Дипломски рад |
| Број линија кода | 673 | 1146 | 1979 | 4085 |
| Број фајлова | 8 | 10 | 21 | 34 |
| Број класа | 11 | 14 | 25 | 42 |
| Број метода | 43 | 57 | 106 | 247 |
| Величина извршног фајла | 766 KB | 777 KB | 795 KB | 26,2 MB |
| Величина *NetBeans* пројекта | 2,81 MB | 2,86 MB | 2,94 MB | 81,7 MB |

Табела 7-1

Из Табеле 7-1 се може видети да је број линија кода контролне лабораторијске вежбе скоро дупло већи од скелета, а код домаћег задатка тај број је приближно утростручен. На нивоу дипломског рада има више него дупло линија кода у односу на домаћи задатак. Несразмеран скок у величини извршног фајла и пројекта код дипломског рада у односу на претходне нивое је објашњив тиме што су ту присутни најпре звучни записи, а потом и већи број слика, који заузимају много више простора него изворни (у саставу *NetBeans* пројекта), односно извршни код.

## 7.2 Занимљиви детаљи имплементације

Овде ће бити наведено пар детаља имплементације решења који читаоцу могу бити занимљиви.

### 7.2.1 Како је пројекат започет

Аутор је првобитно почео с развојем игре *Arena* за потребе демонстратуре за предмет Рачунарска графика у школској 2018/19. години. Као основа пројекта послужило је ауторово решење домаћег задатка из школске 2017/18. године, *Моја трпезарија*, на тему образовног софтвера за помоћ у стицању вештине сналажења у кухињи и трпезарији (текст и изворни код приложени међу артефактима). Нацрт класа Game и Player најпре су рађени по узору на истоимене класе из поменутог решења, а сличности су уочљиве: основна структура кода, детекција колизија, руковање догађајима, кретање играча и друго.

### 7.2.2 Перспектива из првог лица

За окретање камере (погледа играча, у класи Player) помоћу миша, било је једноставно реализовати хоризонталну компоненту, јер је потребно променити само угао објекта типа Rotate који утиче на камеру, при чему је подешено да је оса ротације тог објекта Y (*Arena* се одиграва у једној хоризонталној равни, а Y оса је нормална на њу). Међутим, вертикалну компоненту је било потребно реализовати тако да буде прилагођена тренутној оријентацији камере (може се представити као ротација око X осе фиксиране на играчевом екрану, односно померање погледа горе или доле независно од хоризонталне оријентације). Ово је решено следећим кодом:

upDownRotation.setAxis(

new Point3D(Math.cos(horizontalAngle \* Math.PI / 180),

0,

-Math.sin(horizontalAngle \* Math.PI / 180))

);

где је upDownRotation објекат типа Rotate, a horizontalAngle угао који се срачуна након померања миша горе или доле.

### 7.2.3 *Stream*

У просторији са пројектилима потребно је да редови плоча постану црвени као упозорење пред лансирање пројектила. Због конструктора који су на располагању за класу KeyFrame, било је неопходно као један од параметара проследити својства њихових боја (ObjectProperty<Color>) као низ објеката типа KeyValue. Ту су се корисним показале могућности *Java 8* и могућности које нуди кроз класе из пакета java.util.stream [12]. Позивањем статичке методе stream() класа које је поседују, односно истоимене нестатичке методе објеката који је поседују, враћа се објекат класе која имплементира интерфејс Stream. Тај интерфејс апстрахује секвенцу елемената над којом се могу вршити секвенцијалне и паралелне агрегатне функције помоћу његових метода, као што су toArray(), forEach() и друге. Метода map() објеката класа које имплементирају интерфејс Stream сваки елемент из секвенце појединачно мапира у неки други објекат (потенцијално другог типа) функцијом за мапирање прослеђеном као параметар методе, тако да новонастали Stream представља секвенцу тих нових објеката. Следећим кодом је решен проблем наведен горе:

Arrays

.stream(tiles)

.map(col -> col[zIndex])

.map(tile -> new KeyValue(

((PhongMaterial) tile.getMaterial())

.diffuseColorProperty(),

Color.RED, Interpolator.LINEAR))

.toArray(KeyValue[]::new)

где су tiles плоче на које се дели тло просторије, типа Box[][], а zIndex индекс реда који треба да поцрвени. Специфицирањем првог индекса дводимензионалног низа tiles одређује се колона, а другим и појединачна плоча у датом реду, те се с тим на уму позивом статичке методе Arrays.stream() са параметром tiles добија секвенца колона плоча, а у првом позиву map() свака колона мапира у плочу те колоне, и то са истим индексом (zIndex), што представља ред који треба да поцрвени. Затим се свака од тих плоча другим позивом методе map() мапира у себи одговарајући нов објекат типа KeyValue тако што се за сваку плочу из реда позива конструктор класе KeyValue ком је као први параметар неопходно проследити објекат типа WritableValue, који је у овом случају добијен помоћу те плоче (tile) кроз ((PhongMaterial) tile.getMaterial()).diffuseColorProperty(). Напослетку се добијена секвенца објеката типа KeyValue претвара у низ позивом toArray().

# 8. Закључак

Искуство израде овог дипломског рада несумњиво иде у прилог констатацији да је развој 3Д видео-игре захтеван посао. Потребно је уложити доста времена и труда да би се дизајнирала сцена, креирао сценарио игре, а затим осмислио и написао лепо организован код, који је одржив и проширив. У конкретном случају еволутивног развоја игре за потребе најпре лабораторијских вежби, а затим домаћег задатка (пројекта), да би се на крају надоградила кроз дипломски рад, још више је наглашена потреба за одрживошћу и проширивошћу кода. Поред тога, потребно је имати добру меру за комплексност ових корака у развоју, јер су предвиђена времена за израду појединих корака у решавању ограничена.

Ограничење које је присутно за постизање спектакуларнијих резултата јесте сама технологија: *JavaFX*. Иако веома практична за академске потребе, нема могућности за лако постизање визуелно атрактивних резултата. Међутим, програмерски изазови не зависе од тога. Како су лабораторијске вежбе и домаћи задатак из *Рачунарске графике* практични програмерски изазови, ниво подршке коју програмер добија кроз библиотеку добро је примерена наставним потребама предмета.

Напослетку, једна субјективна примедба аутора. Развој игре се показао пријатно изазовним по свим нивоима и захтева забавну дозу креативности у решавању проблема. Уметнички аспект дизајнирања саме игре и слобода осмишљавања искуства које пружа даје изузетну мотивацију за рад. Такође, имплементирање замисли уме да буде проблематично тамо где се најмање очекује, али се с праксом и навикавањем на коришћени алат отвара много опција – за скоро сваки изазов који стане на пут вероватно је да постоји више решења.

# 9. Литература

1. Hughes, J. F., Van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., Akeley, K., *Computer Graphics: Principles and Practice* (Third Edition), Addison-Wesley, 1995.
2. Sharan, K., *Learn JavaFX 8*, Apress, 2015.
3. *NetBeans* развојно окружење, званична веб-страница: <https://netbeans.org/>, приступано март 2020.
4. *Video game*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Video_game>, приступано март 2020.
5. *Action game*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Action_game>, приступано март 2020.
6. *Platform game*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Platform_game>, приступано март 2020.
7. *First-person (video games)*, Википедија веб-страница: <https://en.wikipedia.org/wiki/First-person_(video_games)>, приступано март 2020.
8. *Rayman 2: The Great Escape*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Rayman_2:_The_Great_Escape>, приступано март 2020.
9. *Croc 2*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Croc_2>, приступано март 2020.
10. *Half-Life*, Википедија веб-страница: <https://en.wikipedia.org/wiki/Half-Life_(video_game)>, приступано март 2020.
11. *SourceMonitor* алат за метрику кода, званична веб-страница: <http://www.campwoodsw.com/sourcemonitor.html>, приступано март 2020.
12. *Package java.util.stream*, званична документација: <https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html>, приступано март 2020.