Inkrementalni razvoj 3D video-igre *Arena* kroz praktičan rad studenata

Vojislav Bogosavljević i Igor Tartalja

***Apstrakt*—U radu je opisano iskustvo inkrementalnog razvoja 3D video-igre kroz njenu evoluciju od skeleta koda prikazanog studentima na pokaznoj laboratorijskoj vežbi na predmetu Računarska grafika, preko kontrolne laboratorijske vežbe i kasnijeg domaćeg zadatka (projekta), sve do diplomskog rada studenta. Cilj razvijene igre je da se za ograničeno vreme sakupi što veći broj poena i preživi u areni sa nekoliko prostorija, izbegavanjem prepreka, uz sakupljanje različitih bonusa. Softver je modelovan na standardnom jeziku UML (*Unified Modeling Language*), a implementiran na programskom jeziku *Java*, uz korišćenje bibilioteke *JavaFX*.**

***Ključne reči*—računarska grafika; 3D video-igra; inkrementalni razvoj softvera; laboratorijska vežba; JavaFX.**

I. Uvod

Računarska grafika je nauka i umetnost komunikacije vizuelnim putem pomoću računarskog prikazivača i uređaja za interakciju [1]. Njen cilj je sinteza statičke ili pokretne slike uz interakciju sa korisnikom. Računarska grafika prožima skoro sve oblasti ljudske delatnosti, a izvesno je da će, kako čovečanstvo dalje bude napredovalo u tehnološkom smislu, oslanjanje na računarsku grafiku biti još veće. Pored značajnih domena primene, poput nauke, tehnologije, medicine, obrazovanja, umetnosti i drugih, njena primena koja daje ovoj oblasti i najjači pogon za razvoj je industrija igara. U ovom radu obrađen je inkrementalni razvoj jedne relativno jednostavne 3D video-igre.

Na Elektrotehničnom fakultetu Univerziteta u Beogradu jedan od izbornih predmeta jeste *Računarska grafika*. U okviru tog predmeta se izučavaju osnovni koncepti i dobija teorijska podloga, a takođe se stečena znanja primenjuju i utemeljuju kroz praktične obaveze na predmetu. Ovaj rad predstavlja jednu studiju slučaja inkrementalnog razvoja 3D video-igre kroz praktične obaveze studenata na ovom predmetu.

Na predmetu *Računarska grafika* se za realizaciju praktičnih zadataka koristi programski jezik Java i biblioteka JavaFX, koja omogućava dovoljno jednostavno i efikasno postizanje rezultata u 2D i 3D grafici [2]. Zbog svoje jednostavnosti i prijemčivosti (a bez gubljenja sveobuhvatnosti), prema iskustvima nastavnika na kursu, ovaj jezik sa ovom bibliotekom predstavlja dobar izbor za akademske svrhe. Razvojno okruženje u kom je razvijan softver kroz predispitne obaveze i diplomski rad je integrisano okruženje *NetBeans* [3], koje nudi podršku za *JavaFX*.

U radu je opisan inkrementalni razvoj jedne 3D video-igre kao teme za izradu najpre laboratorijskih vežbi, zatim domaćeg zadatka (projekta) i, na kraju, diplomskog rada. Tema igre je arena sa tri prostorije u kojima igrač nastoji da preživi određeno vreme u borbi sa različitim vrstama prepreka u različitim prostorijama, kako što su šiljci koji iskaču iz tla, projektili koji preleću kroz prostoriju, odnosno sečiva koja padaju sa plafona.

Rad je prvenstveno namenjen čitaocima iz akademskih ustanova, ali opisana iskustva mogu biti od koristi i mlađim razvojnim programerima u industriji igara.

U sledećem poglavlju se opisuje rešavani problem. U trećem poglavlju je dat pregled igara koje su srodne igri *Arena* ili su poslužile kao inspiracija za nju. U četvrtom je sažeta funkcionalna specifikacija kompletne igre. U petom je dat kratak pregled najvažnijih delova arhitekture programa. U šestom poglavlju je opisana implementacija igre, prikazane su tehničke karakteristike, kao i nekoliko interesantnih detalja implementacije. Sedmo poglavlje posvećeno je diskusiji rezultata. U poslednjem, osmom, poglavlju dat je zaključak, sa nekim smernicama za mogući dalji razvoj igre.

II. Problem

Znanja usvojena na predmetu *Računarska grafika* studenti imaju priliku da praktično primene kroz laboratorijske vežbe, zatim domaći zadatak (projekat) i na kraju kroz diplomski ili master rad. Organizuju se četiri vežbe, od kojih su prva i treća pokaznog karaktera, a druga i četvrta kontrolnog; prve dve su vezane za 2D, druge dve za 3D grafiku. Polovinu poena na predmetu studenti stiču kroz kontrolne laboratorijske vežbe i/ili domaći zadatak

Na pokaznim laboratorijskim vežbama studenti se upoznaju sa programom (skeletom koda) koji će se na kontrolnoj vežbi nadograđivati za ocenu. Skelet koda predstavlja jezgro igre, te sadrži osnovne funkcionalnosti objedinjene u smislenu celinu. On usmerava kasnije projektovanje igre, obezbeđujući pri tome visok stepen fleksibilnosti u implementaciji zadataka vezanih za grafičke objekte, njihovu animaciju i interakciju sa korisnikom. Studenti taj skelet proširuju dodatnim funkcionalnostima i kroz to utvrđuju svoja znanja, pokazuju u kojoj su meri savladali materiju i vežbaju da vrše izmene, odnosno proširenja nad tuđim kodom. Na 3. laboratorijskoj vežbi se studenti upoznaju sa skeletom koda i ostavi im se oko 30min za implementaciju neke dodatne funkcionalnosti,

Vojislav Bogosavljević – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar Kralja Aleksandra 73, 11020 Beograd, Srbija (e-mail: vbogos3@gmail.com).

Igor Tartalja – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar Kralja Aleksandra 73, 11020 Beograd, Srbija (e-mail: tartalja@etf.rs).

eventualno uz pomoć demonstratora, što se ne ocenjuje. Na laboratorijskoj vežbi studenti dobijaju veći broj (do 20) zadataka za dodavanje novih funkcionalnosti skeletu koda.

Kao tema treće i četvrte laboratorijske vežbe, prikazanih u ovom radu, odabrana je 3D video-igra. Motivacija za to je činjenica da problem programiranja video-igre obuhvata više različitih aspekata. Najpre, što je od naročitog interesa za predmet *Računarska grafika*, tu su vizuelni dizajn i realizacija grafičkog dela, a zatim osmišljavanje igre i iskustva koje ona pruža (eng. *gameplay*). Dalje, tu je primena postojećih i razvoj novih algoritama za implementaciju delova igre, pisanje kvalitetno strukturiranog, održivog i proširivog koda, te dizajn zvuka (pozadinske muzike i zvučnih efekata) i drugo [4]. Značajna je činjenica da se ishodi razvoja veoma brzo uočavaju, odnosno potrebno je relativno malo vremena i truda da bi se postigli vidljivi rezultati, što može biti veoma motivišuće studentima.

Praktični deo predmeta Računarska grafika se kompletira kroz domaći zadatak (projekat). Ovaj projekat obuhvata dalju nadogradnju implementacije sa laboratorijske vežbe, a projektni zahtevi su osetno složeniji i obimniji. Na kraju, polazeći od nivoa domaćeg zadatka, igra se može dodatno proširiti još kompleksnijim funkcionalnostima, kako bi dostigla nivo diplomskog ili master rada.

III. Srodna rešenja

Arena je žanrovski najbliža opštoj kategoriji akcionih igara (eng. *action game*) u kojima je naglasak na fizičkim izazovima, što uključuje koordinaciju između ruke i oka i brzinu reakcije [5]. Mogla bi se povući i paralela sa platformskim igrama (eng. *platformer*) kao podžanrom akcionih, u kojima je cilj, u opštem slučaju, skakati i penjati se do odvojenih platformi, uz izbegavanje prepreka [6]. Međutim, čest je slučaj da vizura platformske igre bude iz perspektive trećeg lica, što ovde nije slučaj. Perspektiva u igri *Arena* je (primarno) iz prvog lica (eng. *first-person*), rađena najpre po ugledu na "pucačke" igre iz prvog lica [7]. Takođe, ne postoji mogućnost skakanja, pošto se radnja igre odigrava u jednoj horizontalnoj ravni.

Arena nije rađena kao klon neke postojeće igre, ali pojedini aspekti su realizovani po uzoru na druge igre. Igre iz kojih je stigla određena inspiracija navedene su u nastavku.

A. Rayman 2: The Great Escape

Rayman 2: The Great Escape [8] je platformska video-igra u kojoj igrač kontroliše Rejmena, humanoidnog lika karakterističnog po tome što mu šake, stopala i glava lebde odvojeni od torzoa. On ima mogućnost gađanja protivnika energetskim projektilima, a dodatnu kontrolu kretanja prilikom skokova mu dozvoljava pokretanje kose u maniru elise helikoptera. Radnja igre se kreće oko invazije robota-gusara iz svemira koji nameravaju da okupiraju svet. Igru karakterišu kvalitetno dizajnirani nivoi, sveprisutna mistična atmosfera i upečatljivi likovi. Elementi igre koji su poslužili kao inspiracija za *Arenu* jesu raznovrsne prepreke, poput pipaka koji iskaču iz zidova i projektila kojima protivnici gađaju igrača.

B. Croc 2

Croc 2 [9] je platformska video-igra u kojoj igrač kontroliše Kroka, krokodila u potrazi za nestalim roditeljima. Igrač ima razne mogućnosti kontrole, kao što su višestruki skok i različite vrste napada. Svet je dosta otvoren i često je na igraču da odabere kojim će redosledom prelaziti postave. Za ovu igru je karakterističan veliki broj raznovrsnih nivoa i njihov konceptualni i vizuelni dizajn, od kojih su neki elementi poslužili kao uzor za *Arenu*, među kojima su prepreke poput iskačućih šiljaka i srca za sakupljanje, radi obnavljanja izgubljenih života.

C. Half-Life

U pitanju je jedna od najuticajnijih igara iz žanra pucačkih igara iz prvog lica, koja se ubraja među najbolje igre svih vremena [10]. Srž scenarija igre je borba protiv vanzemaljskih i ljudskih protivnika i rešavanje zagonetki. Igrač kontroliše Gordona Frimena, naučnika koji se bori za svoj život nakon što u istraživačkom centru eksperiment sa vanzemaljskim materijalom pođe naopako. Iako *Arena* žanrovski nema pucački karakter, generalni osećaj kontrole iz prvog lica i elementi vizuelnog identiteta (na primer, veličina vidnog polja, brzina kretanja i dizajn šiljaka i sečiva) crpeli su inspiraciju iz igre *Half-Life*; valja naglasiti da ova igra poseduje i platformske elemente.

IV. Funkcionalnosti softvera

Arena je 3D akciona igra u kojoj je cilj igrača da sakupi što više poena za ograničeno vreme, uz izbegavanje prepreka. Iako je primarno igra iz prvog lica, postoji i druga kamera u centru prostorije na vertikalnoj osi, što omogućava pregled svih delova prostorije i pogled na avatara igrača. Glavni meni omogućava korisniku da pokrene igru, vidi tabelu najboljih rezultata, uđe u podešavanja i napusti program. Pre započinjanja igre neophodno je uneti ime igrača.

Igrač upravlja kretanjem svog avatara komandama sa tastature, uključujući mogućnosti hodanja ili trčanja. Pomeranjem miša se usmerava kamera iz prvog lica, odnosno menja orijentacija igrača. Igru je moguće privremeno zaustaviti (pauzirati) i nastaviti.

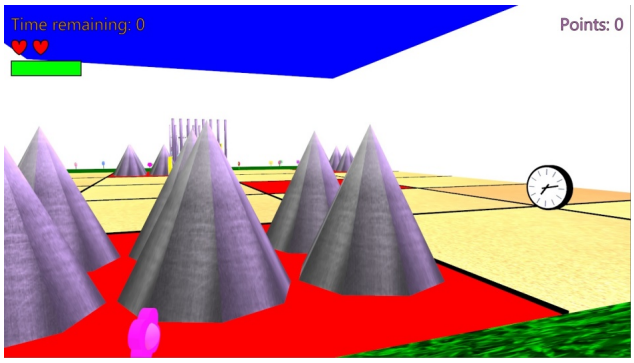
Arena je podeljena na tri prostorije, sa različitim vrstama skrivenih prepreka u svakoj: šiljci (Sl. 1), projektili (Sl. 2) i sečiva (Sl. 3). Pri kontaktu sa njima, igrač gubi jedan život. Tlo prostorija je podeljeno na ploče. Nešto ranije u odnosu na pojavljivanje prepreke, odgovarajuća ploča (u 1. prostoriji), red ploča (u 2. prostoriji) ili grupa susednih ploča (u 3. prostoriji) na kojima će se pojaviti prepreka će pocrveneti sugerišući igraču nastupajuću opasnost. Nad poljima se stvaraju predmeti za sakupljanje: novčići koji daju poene, srca koja obnavljaju živote i satovi koji daju dodatno vreme (na slikama prostorija mogu se videti i ti predmeti).

Igračev interfejs (eng. *Head-up Display*, HUD), stalno prikazan tokom igre u vrhu ekrana, sadrži informacije o preostalom vremenu, životima i energiji za trčanje. Energija se troši dok igrač trči, a dok stoji ili hoda ona se obnavlja.

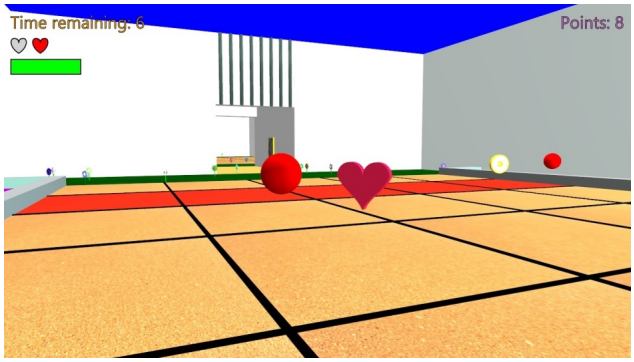
Odbrojavanje vremena u prostoriji počinje čim igrač

zakorači u prostoriju iz hodnika koji joj prethodi. Nakon isteka vremena u jednoj prostoriji, otvaraju se kapije hodnika ka narednoj.

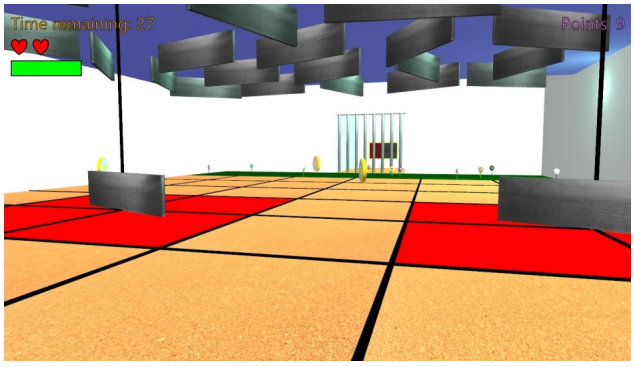
Igra se uspešno završava ukoliko igrač dočeka istek vremena u poslednjoj prostoriji, a neuspešno ako izgubi sve živote. Bolje je plasiran onaj igrač koji je sakupio više poena. Detalji igre opisani su u dokumentu Prilog A (videti Dodatak).



Sl. 1. Prva prostorija sa preprekama – šiljcima



Sl. 2. Druga prostorija sa preprekama – projektilima

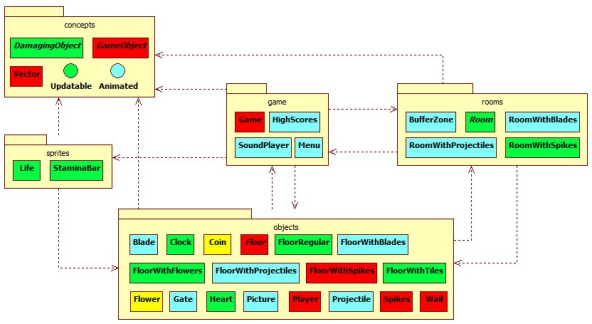


Sl. 3. Treća prostorija sa preprekama – sečivima

V. Projekat softvera

Na Sl. 4 prikazan je UML dijagram paketa sa svim klasama i interfejsima u projektu. Crvenom bojom označene su klase iz skeleta koda, odnosno sa pokazne vežbe, žutom klase dodate na kontrolnoj laboratorijskoj vežbi, zelenom klase dodate kroz domaći zadatak, a tirkiznom klase dodate kroz diplomski rad. Detalji projekta softvera opisani su u dokumentu Prilog B (videti Dodatak).

U paketu `concepts` nalaze se klase i interfejsi koji su osnova za veliki broj ostalih klasa: `GameObject` je natklasa svih klasa u paketima `objects` i `rooms`, `Vector` je prisutan kao polje u klasi `GameObject` i kao parametar njenog konstruktora, dok interfejse `Updatable` i `Animated` implementiraju mnoge klase iz navedenih paketa, kao i paketa `sprites`.



Sl. 4. UML dijagram paketa razvijenog softvera

Klase koje predstavljaju opštije celine i pokretače dešavanja u igri nalaze se u paketu `game`. Među njima je glavna klasa `Game` u kojoj je i `main()` metod. Ta klasa je ključna za tok igre, omogućava pauziranje i prelazak u meni, a takođe poseduje tajmer na čije se otkucaje odigravaju mnoge važne stvari, poput izazivanja događaja u određenim trenucima, detekcije kolizija, računanje protoka vremena i slično. Praktično povezuje sve klase u projektu.

Paket `objects` sadrži 3D objekte koji se pojavljuju u igri, kao što su satovi, tlo, zidovi i slično. Tu je i klasa `Player` koja ima nešto složenije mogućnosti: predstavlja igrača, ima nosač za kameru za pogled iz prvog lica, rukovaoce događajima za korisnički ulaz radi kontrole avatara i vrši značajan deo upravljanja kretanjem po areni.

Iako predstavljaju 3D objekte, odnosno prostorije u areni, klase unutar paketa `rooms` su u njega izdvojene iz razloga što su u pitanju objekti sastavljeni od većeg broja jednostavnijih, iz paketa `objects`. Takođe, ove klase realizuju funkcionalnosti koje utiču na dešavanja u igri i nisu samo statične celine koje služe da budu korišćene od strane drugih delova koda (kao većina klasa u paketu `objects`).

U `sprites` su smešteni 2D objekti koji su deo korisničkog interfejsa – srca koja predstavljaju preostale živote i pokazivač preostale energije za trčanje.

VI. Implementacija softvera

Osnova za implementaciju igre je skelet koda, koji se stavlja studentima na raspolaganje. Polazeći od njega, igra se proširuje do nivoa kontrolne laboratorijske vežbe, na kojoj studenti samostalno realizuju dodatne funkcionalne zahteve. Te izmene su prilagođene ograničenom vremenu izrade laboratorijske vežbe, te su mahom estetske prirode, sa izuzetkom dodavanja novčića, čime se igri nazire poenta. Odatle, realizacijom stavki za domaći zadatak, aplikacija počinje da poprima opšte crte poznate za veliki broj igara: ograničeno vreme za sakupljanje što većeg broja poena i

mogućnost gubitka života čine igru izazovnom i otvaraju mogućnost postizanja uporedivih rezultata po odigranim partijama, dok postojanje obnovljivih zaliha energije za trčanje i pojava satova za dodatno vreme navodi igrača da se pametno kreće po areni. Nadogradnja od nivoa domaćeg zadatka do diplomskog rada kompletira video-igru: više prostorija daje veću slobodu kretanja i predstavlja mali svet za sebe, pozadinska muzika i zvučni efekti doprinose atmosferi, a postojanje menija i mogućnost pauziranja igre nagoveštavaju da je u pitanju celovito rešenje, nalik modernim igrama. U nastavku će biti ukratko opisani neki interesantni elementi implementacije.

A. Implementacija pogleda iz prvog lica

Za okretanje kamere (pogleda igrača), pomoću miša, bilo je jednostavno u klasi `Player` realizovati rotaciju u horizontalnoj ravni (pri pomeranju miša levo ili desno), jer je potrebno promeniti samo ugao objekta bibliotečke klase `Rotate` zaduženog za rotaciju u toj ravni. Osa rotacije je podešena da bude Y osa, pošto se igra odvija u jednoj horizontalnoj ravni, a Y osa je normalna na nju. Osa rotacije u vertikalnoj ravni se podešava sledećim kodom:

```
upDownRotation.setAxis(
  new Point3D(
    Math.cos(horizontalAngle*Math.PI/180),
    0,
    -Math.sin(horizontalAngle*Math.PI/180)
  )
);
```

gde je `upDownRotation` objekat tipa `Rotate`, zadužen za rotaciju u vertikalnoj ravni (pri pomeranju miša gore ili dole), a `horizontalAngle` ugao rotacije u horizontalnoj ravni, koji se izračuna nakon pomeranja miša levo ili desno. Tako postavljena osa rotacije predstavlja horizontalnu X osu igrača. Kasnijim pozivom metode `setAngle()` objekta `upDownRotation` kamera se rotira u vertikalnoj ravni za prosleđenu vrednost ugla `verticalAngle` izračunatu nakon pomeranja miša gore ili dole.

B. Implementacija upozoravajućeg bojenja reda ploča

U prostoriji sa projektilima potrebno je da redovi ploča postanu crveni kao upozorenje pred lansiranje projektila. Zbog konstruktora koji su na raspolaganju za bibliotečku klasu `KeyFrame`, bilo je neophodno kao jedan od parametara proslediti svojstva njihovih boja (`ObjectProperty<Color>`) kao niz objekata tipa `KeyValue`. Tu su se korisnim pokazale mogućnosti Java 8 i mogućnosti koje nudi kroz klase iz paketa `java.util.stream` [11]. Pozivanjem statičkih metoda `stream()` klasa koje ga poseduju, odnosno istoimenog nestatičkog metoda objekata koji ga poseduju, vraća se objekat klase koja implementira interfejs `Stream`. Taj interfejs apstrahuje sekvencu elemenata nad kojom se mogu vršiti sekvencijalne i paralelne agregatne funkcije pomoću njegovih metoda, kao što su `toArray()`, `forEach()` i druge. Metod `map()` objekata klasa koje implementiraju interfejs `Stream` svaki element iz sekvence pojedinačno preslikava u neki drugi objekat (potencijalno drugog tipa) funkcijom prosleđenom kao parametar metoda, tako da novonastali objekat tipa `Stream` predstavlja sekvencu tih novih objekata. Sledećim kodom je rešen navedeni problem:

```
Arrays
    .stream(tiles)
    .map(col -> col[zIndex])
    .map(tile -> new KeyValue(
        ((PhongMaterial)tile.getMaterial())
            .diffuseColorProperty(),
        Color.RED,
        Interpolator.LINEAR))
    .toArray(KeyValue[]::new)
```

gde su `tiles` ploče na koje se deli tlo prostorije, tipa `Box[][]`, a `zIndex` indeks reda koji treba da pocrveni. Specificiranjem prvog indeksa dvodimenzionalnog niza `tiles` određuje se kolona, a drugim i pojedinačna ploča u datom redu. Imajući to u vidu, pozivom statičkog metoda `Arrays.stream(tiles)` dobija se sekvenca kolona ploča, a u prvom pozivu `map()` svaka kolona se preslikava u ploču te kolone, i to sa istim indeksom (`zIndex`), što predstavlja red koji treba da pocrveni. Zatim se svaka od tih ploča drugim pozivom metoda `map()` preslikava u sebi odgovarajući nov objekat tipa `KeyValue` tako što se za svaku ploču iz reda poziva konstruktor klase `KeyValue`, kom se kao prvi parametar prosleđuje difuzna boja date ploče, kao drugi crvena boja, a kao treći objekat linearnog interpolatora. Naposletku se dobijena sekvenca objekata tipa `KeyValue` pretvara u niz pozivom metoda `toArray()`.

C. Tehničke karakteristike softvera

U Tabeli 1 prikazane su tehničke karakteristike rešenja igre po različitim fazama razvoja. Za analizu je korišćen program *SourceMonitor* [12].

TABELA I
TEHNIČKE KARAKTERISTIKE IMPLEMENTACIJE IGRE *ARENA* PO FAZAMA

	Skelet koda	Kontrolna lab vežba	Domaći zadatak	Diplomski rad
Broj linija koda	673	1146	1979	4085
Broj fajlova	8	10	21	34
Broj klasa	11	14	25	42
Broj metoda	43	57	106	247
Veličina izvršnog fajla	766 KB	777 KB	795 KB	26,2 MB
Veličina *NetBeans* projekta	2,81 MB	2,86 MB	2,94 MB	81,7 MB

Iz Tabele 1 se može uočiti da je broj linija koda kontrolne laboratorijske vežbe skoro duplo veći od broja linija skeleta koda, a kod domaćeg zadatka taj broj je približno utrostručen. Na nivou diplomskog rada ima više nego duplo linija koda u odnosu na domaći zadatak. Nesrazmeran skok u veličini izvršnog fajla i projekta kod diplomskog rada u odnosu na prethodne nivoe je objašnjiv time što su tu prisutni najpre zvučni zapisi, a potom i veći broj slika, koji zauzimaju mnogo više prostora nego izvorni (u sastavu *NetBeans* projekta), odnosno izvršni kod.

VII. DISKUSIJA

Opisane laboratorijske vežbe sa domaćim zadatkom sprovedene su školske 2018/19 godine. U Tabeli 2 navedeni su podaci o broju studenata, prosečnoj oceni (aritmetičkoj sredini) i standardnoj devijaciji ocene na kontrolnoj laboratorijskoj vežbi, domaćem zadatku i diplomskom radu. Ukupan broj studenata upisanih na predmet *Računarska grafika*, koji su pristupili barem jednoj predispitnoj obavezi, bio je 35. Na kontrolnoj laboratorijskoj vežbi studenti su rešavali 17 zadataka od kojih je svaki nosio od 5 do 20 poena. U zbiru, svi zadaci su nosili 150 poena, pri čemu je broj poena koji nosi vežba ograničen na 100, tako da za maksimalan broj poena na vežbi nije bilo potebno realizovati sve zadatke. Laboratorijska vežba je trajala 120 min. S obzirom da je procenjena razlika u broju linija koda softvera razvijenog na kontrolnoj laboratorijskoj vežbi (kada je realizovano svih 17 zadataka) i skeleta koda koji se studentima stavlja na raspolaganje, nešto manje od 500 linija koda, za ostvarivanje maksimalnog broja poena studentima je stavljeno u zadatak da za 120 minuta za realizuju nešto više od 300 linija koda za maksimalni broj poena, što svakako zahteva visok stepen znanja i veštine programiranja 3D grafike.

TABELA II
STATISTIKA PREDISPITNIH OBAVEZA I DIPLOMSKIH RADOVA

	Broj studenata	Prosečna ocena	Standardna devijacija
Kontrolna lab vežba	26	53.78	30.20
Domaći zadatak	24	94.50	12.47
Diplomski rad	4	10	0

Pri analizi prikazanih rezultata treba imati u vidu činjenicu da poeni sa domaćeg zadatka koji se brani u prvom ispitnom roku, a koji predstavlja nadgradnju video-igre sa kontrolne laboratorijske vežbe, mogu da kompenziju poene sa te laboratorijske vežbe. To je jedan od razloga relativno niske prosečne ocene i velike standardne devijacije ocene na laboratorijskoj vežbi. Drugi razlog relativno male prosečne ocene je ograničeno vreme vežbe u kojem je potrebno realizovati veći broj netrivijalnih zadataka vezanih za funkcionalnosti i grafički dizajn igre.

S obzirom na činjenicu da se poeni sa laboratorijske vežbe mogu kompenzovati poenima osvojenim kroz domaći zadatak, kao i činjenicu da nije potrebno realizovati sve zadatke za maksimalni broj poena, već student može da izabere podskup zadataka koje će realizovati, nije od presudnog značaja da se broj i težina zadataka precizno prilagode raspoloživom vremenu za njihovo rešavanje. Činjenica da su dva od 26 studenata osvojila 99% i 100% poena na laboratorijskoj vežbi, pokazuje da je raspoloživo vreme bilo dovoljno za realizaciju potrebnog broja zadataka i osvajanje maksimalnog broja poena. Ključnu ulogu kontrolna laboratorijska vežba odigrala je kroz stimulisanje oko dve trećine studenata da blagovremeno počnu sa proučavanjem gradiva iz 3D grafike, provere svoje znanje i veštinu programiranja kroz izradu laboratorijske vežbe, kako bi bolje pripremljeni, ubrzo posle sprovedene vežbe, započeli rad na relativno kompleksnom domaćem zadatku i odbranili ga u prvom ispitnom roku. Izuzetno visoka prosečna ocena i relativno mala standardna devijacija ocene domaćih zadataka 24 studenta, od 26 koji su radili kontrolnu laboratorijsku vežbu, ide u prilog navedenoj pretpostavci o ključnoj ulozi laboratorijske vežbe u motivaciji učenika.

VIII. ZAKLJUČAK

Iskustvo izrade ovog rada nesumnjivo ide u prilog konstataciji da je razvoj 3D video-igre zahtevan posao. Potrebno je uložiti značajno vreme i trud da bi se dizajnirala scena, kreirao scenario igre, a zatim osmislio i napisao lepo organizovan kod, koji je održiv i proširiv. U konkretnom slučaju evolutivnog razvoja igre za potrebe najpre laboratorijskih vežbi, a zatim domaćeg zadatka (projekta), da bi se na kraju nadogradila kroz diplomski rad, još više je naglašena potreba za održivošću i proširivošću koda. Pored toga, potrebno je imati dobru meru za kompleksnost ovih koraka u razvoju, jer je predviđeno vreme za izradu pojedinih koraka u rešavanju ograničeno.

Ograničenje koje je prisutno za postizanje spektakularnijih rezultata jeste sama tehnologija oslonjena na biblioteku *JavaFX*. Iako veoma praktična za akademske potrebe, ova biblioteka nema mogućnosti za lako postizanje vizuelno atraktivnih rezultata. Međutim, programerski izazovi ne zavise od toga. Kako su laboratorijske vežbe i domaći zadatak iz *Računarske grafike* praktični programerski izazovi, nivo podrške koju programer dobija kroz biblioteku *JavaFX* dobro je primeren nastavnim potrebama predmeta.

Prema iskustvu prvog autora, koji je razvio 3D video-igru *Arena* kroz svoj diplomski rad, taj razvoj se pokazao izazovnim, zahtevajući zabavnu dozu kreativnosti u rešavanju problema. Aspekti dizajniranja scene i scenarija igre uz slobodu u osmišljavanju iskustva koje igra pruža igraču, dali su izuzetnu motivaciju za razvoj pokazne i kontrolne laboratorijske vežbe, domaćeg zadatka na predmetu *Računarska grafika* i, na kraju, softvera ciljne igre realizovanog kroz diplomski rad.

Iako je realizovana igra kroz ovaj rad potpuno funkcionalna, ona je samo praktičan rezultat diplomskog rada i može se dalje nadograđivati po mnogim aspektima, sa ciljem da dostigne nivo master rada ili čak produkcioni nivo. Najpre, dizajn igre je relativno skroman i mogao bi se značajno unaprediti korišćenjem realističnijih 3D modela objekata i maštovitijih tekstura. Zatim bi se, po ugledu na mnoge akcione igre, mogli uključiti i protivnici u prostorijama, koji vođeni veštačkom inteligencijom beže od igrača i napadaju ga, pucajući na njega, dok im igrač uzvraća vatrom iz različitih oružja. Pored ovakvih dodatnih funkcionalnosti, igra bi se mogla proširiti kvizom znanja, tako što igrač po kompletiranju jednog nivoa (po završenom boravku u jednoj prostoriji) mora da reši kratak kviz znanja da bi prešao na sledeći nivo, odnosno ušao u sledeću prostoriju. I u toku boravka u jednoj prostoriji, mogli bi se pojavljivati novi predmeti koji nose veći broj poena igraču kada ih sakupi i odgovori na pitanje koje mu se tom prilikom postavi. Na taj način bi zabavna igra, realizovana u ovom radu, prerasla u

obrazovnu igru upotrebljivu na različitim nivoima obrazovanja i bez ograničenja u oblasti obrazovanja.

DODATAK

Kompletno uputstvo za korišćenje sa detaljnim opisom funkcionalnosti igre (Prilog A), celovit UML model igre (Prilog B) i izvršna verzija igre *Arena*, mogu se preuzeti sa adrese: https://www.dropbox.com/sh/4ggphvj83owcfvg/AAArlFNMO8mt74quPi4fxZgra?dl=0

Postavke zadataka za laboratorijske vežbe, domaćeg zadatka (projekta) sa dopunskim zahtevima za diplomski i master rad, kao i izvršne verzije skeleta i implementiranih programa sa laboratorijskih vežbi, mogu se preuzeti sa adrese: https://www.dropbox.com/sh/ooso0olzky7jy65/AADrAqLJ5r_n5f_LBRCyDI5Xa?dl=0

ZAHVALNICA

Igor Tartalja je angažovan na projektima TR32039 i TR32047 finansiranim od strane Ministarstva prosvete, nauke i tehnološkog razvoja.

LITERATURA

[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: Building User Experience and Interfaces with Java 8, Apress, 2015.
[3] NetBeans razvojno okruženje, zvanična veb stranica: https://netbeans.org/, pristupano – jul 2020.
[4] Video game, Wikipedia, veb stranica: https://en.wikipedia.org/wiki/Video_game, pristupano – jul 2020.
[5] Action game,Wikipedia, veb stranica: https://en.wikipedia.org/wiki/Action_game, pristupano – jul 2020.
[6] Platform game, Wikipedia, veb stranica: https://en.wikipedia.org/wiki/Platform_game, pristupano – jul 2020.
[7] First-person (video games), Wikipedia, veb stranica: https://en.wikipedia.org/wiki/First-person_(video_games), pristupano – jul 2020.
[8] Rayman 2: The Great Escape: https://store.ubi.com/ie/game?pid=56c4947e88a7e300458b465c&dwvar_56c4947e88a7e300458b465c_Platform=pcdl&edition=Standard%20Edition&source=detail, pristupano – septembar 2020.
[9] Croc 2, Wikipedia, veb stranica: https://en.wikipedia.org/wiki/Croc_2, pristupano – jul 2020.
[10] Half-Life: https://www.half-life.com/en/halflife, pristupano – septembar 2020.
[11] Package java.util.stream, zvanična dokumentacija: https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html, pristupano – jul 2020.
[12] SourceMonitor, alat za metriku koda, zvanična veb-stranica: http://www.campwoodsw.com/sourcemonitor.html, pristupano – jul 2020.

ABSTRACT

The paper describes an experience of incremental development of a 3D video-game through its evolution from the code skeleton presented to students at a tutorial laboratory exercise in the Computer Graphics course, through a control laboratory exercise and later homework (project), all the way to the student 's diploma thesis. The goal of the developed game is to collect as many points as possible in a limited time and survive in an arena with several rooms, avoiding obstacles, while collecting various bonuses. The software is modeled in the standard UML language (*Unified Modeling Language*), and implemented in the *Java* programming language, using the *JavaFX* library.

Incremental development of the 3D video-game *Arena* through students' practical work

Vojislav Bogosavljević, Igor Tartalja