SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 663

Mobilna igra za vježbanje matematike

Denis Pipalović

Zagreb, svibanj 2022.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Sadržaj

1. Uvod

Matematika, osim što je predmet koji se predaje u svim razredima osnovnih i srednjih škola, također je i znanost iz područja prirodnih znanosti. Značaj matematike vidljiv je ne samo u drugim poljima područja prirodnih znanosti, već i u svim drugim područjima, posebice u području tehničkih znanosti. Ne poznavanje osnova matematike negativno se odražava na učenikove školske uspjehe kroz cijelo školovanje. Jedan od problema svakako je problem pristupa nastavnika u učenju matematike, odnosno manjak povlačenja paralela sa stvarnim svijetom i pokazivanja primjena matematike; ovakav pristup demotivira učenika u vježbanju matematike i savladavanju čak i osnovnih računskih operacija. Jednom izgubljenu motivaciju za neko područje teško je vratiti nazad.

Primjenom gamifikacije u edukaciji učenicima bi se razne teme i ishodi učenja mogle prezentirati kroz zanimljive načine, samim time i povećati zainteresiranost za nekom temom ili predmetom. S obzirom na to da djeca u novije doba koriste pametne uređaje sve više i više, gamifikacija jedan od način kroz koji bi to vrijeme na mobitelu moglo biti potrošeno u edukativne svrhe; kroz zabavu!

S obzirom na navedeno, tema ovog rada je napraviti mobilnu igru za vježbanje matematike u nižim razredima osnovne škole koja će kroz zanimljiv i interaktivan način pružati ugodnije iskustvo vježbanja matematičkih zadataka. Igra je zamišljena kao tzv. „endless runner“ u kojoj igrač skuplja padajuće objekte ovisno o trenutnom zadatku (npr. samo parne brojeve, brojeve veće od zadanog, brojeve djeljive s određenim brojem, samo kvadrate i slično) i sakuplja bodove. Kako igra ne bi postala monotona kroz kratko vrijeme igranja cilj je napraviti da igra postaje progresivno sve teža i teža, no do te mjere da se ne postigne negativan efekt u drugom ekstremu; igra ne smije postati niti previše teška jer bi samim time postala i demotivirajuća! Stoga je cilj da se težina igre dinamički prilagođava mogućnostima učenika. Osim same mobilne

igre, potrebno je izgraditi i jednostavno web sučelje za učitelja kako bi mogao postavljati tipove zadataka koji prate temu onoga što se trenutno predaje na nastavi. Kroz navedeno web sučelje nastavniku će se pružati i mogućnost pregleda rezultata pojedinog učenika te uvid u detalje igre kao što je tip zadatak na kojem je učenik griješio, ukupno vrijeme igranja i slično. Kako bi igra bila dostupna svima koji ju požele igrati, odnosno kako igra ne bi ovisila o postavkama učitelja, postojati će i zadane postavke koje će se moći prilagođavati kroz postavke same igre.

2. Naslov 1

Poglavlje 1

2.1. Introduction

Podpoglavlje 1.1

2.2. Introduction

Podpoglavlje 1.2

3. Implementacija mobilne igre

3.1. Pozadina

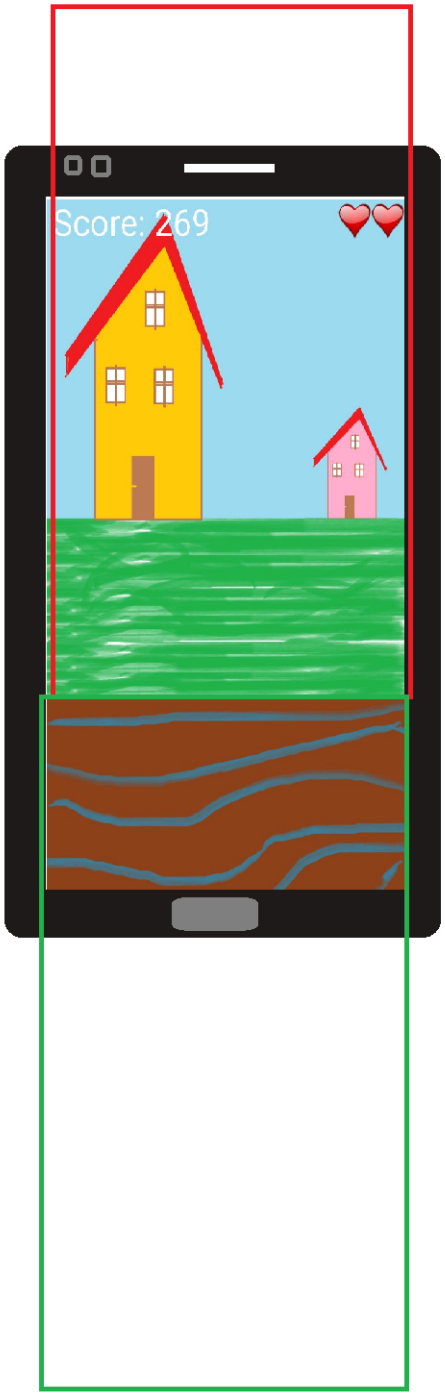
Pozadina je tematski zamišljena kao put iz središta zemlje do dubokog svemira. Mobilna igra je takozvani endless runner, što znači da se potencijalno može igrati beskonačno, odnosno ne postoji način na koji igra završava osim u slučaja da igrač izgubi sve živote, stoga je osim samog efekta "putovanja ka svemiru" potrebno osigurati i beskonačno pomicanje same pozadine.

Pozadina kao takva se u konačnici sastoji od 20ak slika čije spajanje vertikalno daje efekt kontinuiranosti. Dok poznati pokretački strojevi (engl. *game engine*) poput Unity nudi opcije manipulacija pozadinom, kao što je beskonačno ponavljajuća pozadina, ta mogućnost u ovakvom razvoju aplikacije nije moguća, već ju je potrebno iskodirati samostalno. Za početak je potrebno učitati slike pozadina kao Bitmap te ih omotati u objekte kako bi im se dala različita svojstva poput x i y koordinata gdje se trenutno nalaze. Takvi objekti pohranjeni su u polje.

Radi jednostavnosti, svaka slika pozadine se vertikalno i horizontalno skalira na broj piksela koje zaslon mobitela ima; zbog toga, u jednom trenutku na zaslonu se mogu vidjeti maksimalno dvije slike pozadine od jednom. Na samom početku igre uzimaju se prva dva objekta pozadine iz polja. Prvi objekt predstavlja donju pozadinu koja će biti prikazana, a drugi gornju pozadinu. Donju pozadinu potrebno je inicijalizirati na lokaciju sa x koordinatom 0 i y koordinatom 0, odnosno želimo da se pokazuje od početka zaslona vertikalno i horizontalno. Drugu sliku inicijaliziramo opet tako da joj damo x koordinatu 0, no y koordinatu postavljamo na minus vertikalnu veličinu zaslona. Na svako ponovno osvježavanje igre, pozadina se vertikalno spušta ka dolje čime se stvara efekt putovanja igrača prema gore. Spuštanje pozadine prema dolje implementacijski je ostvareno tako što se svakim osvježavanjem zaslona povećava y koordinata obje slike za određenu vrijednost, ovisno o tome kojom

brzinom želimo da "igrač putuje ka svemiru".

Na sljedećoj slici može se vidjeti način iscrtavanja pozadine na ekranu i izvan njega.



Slika 3.1: Pomicanje pozadine

U trenutku kada y koordinata donje slike izađe izvan ekrana (odnosno kada

je vrijednost y osi veća od y dimenzije ekrana) nova donja slika sada je slika koja je bila gore, a novu gornju sliku uzimamo kao sljedeći član polja. Zadnja slika prikazuje svemir pun zvijezdi i nju u polje učitavamo dva puta (dva različita objekta). U trenutku kad smo u igri došli do te slike, tim dvijema slikama se konstanto mijenjaju vrijednosti y osi. Odnosno kad donja slika izađe dolje s ekrana, ponovno se postavljaju vrijednosti y osi za obje slike.

Cijeli način implementacije pomicanja pozadine možemo prikazati sljedećim kodom:

```
1   private void updateBackground() {
2        currentDownBackground.setY(currentDownBackground.getY() + 5);
3        currentUpBackground.setY(currentUpBackground.getY() + 5);
4
5        if (currentDownBackground.getY() > screenY) {
6            currentDownBackground = currentUpBackground;
7            if (currentBackgroundIndex < backgrounds.length - 1)
8                currentUpBackground = backgrounds[++currentBackgroundIndex];
9            else {
10               currentDownBackground = backgrounds[backgrounds.length - 2];
11               currentDownBackground.setY(0);
12               currentUpBackground = backgrounds[backgrounds.length - 1];
13           }
14           currentUpBackground.setY(-screenY);
15       }
16    }
```

3.2. Objekt za skupljanje

Slično kao i pozadinu, objekt koji se kontrolira, odnosno objek s kojim se skupljaju padajući objekt učitan je kao kao Bitmap, odnosno više njih, konkretnije kao dva Bitmap objekta omotana sa objektom klase "Saw". Za razliku od pozadine koja je putovala prema dolje, objekt s kojim skupljamo padajuće objekte je statičan na ekranu, odnosno ne mijenja svoj položaj na ekranu bez korisničke akcije. Pod kornisničku akciju u ovom slučaju podrazumijeva se dodir ekrana. Koristeći oblikovni obrazac promatrač (engl. *observer*), pri inicijalizaciji igre subjektu (objektu konkretne klase "View", u ovom slučaju to je "SurfaceView") se postavlja konkretna implementacija apstraktnog promatrača "OnTouchListener". Navedeno sučelje sastoji se od jedne funkcije:

```
boolean onTouch(View v, MotionEvent event);
```

Ova funkcija prima dva parametra, "View v" koji je pokazivač na "View" kojem je događaj dodira dostavljen te parametar "MotionEvent event" koji opi-

suje tip događaja. Primjeri tipova događaja su: "ACTION_DOWN" koji govori da se dogodio pritisak na zaslonu, "ACTION_UP" koji govori da je dodir zaslona otpušten, "ACTION_MOVE" koji govori da se dogodila promjena pozicije između pritiska ("ACTION_DOWN") i otpuštanja ("ACTION_UP") te nekolicina raznih drugih akcija. Navedena funkcija vraća primitivni tip boolean koji u ovom slučaju predstavlja zastavicu koja govori da li je konkretni promatrač "konzumirao događaj do kraja".

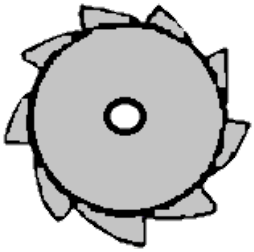
U nastavku je prikazan cijeli programski kod potreban za pomicanje objekta za skupljanje. Konkretna implementacija sučelja "OnTouchListener" izvedena je kao anonimni razred, odnosno zbog bolje preglednosti koda koristi se pokrata u vidu lambda izraza, dok "this" predstavlja objekt klase View.

```
1            this.setOnTouchListener((v, event) -> {
2                if (event.getAction() == MotionEvent.ACTION_MOVE) {
3                    saw.setX((int) event.getX());
4                }
5                return true;
6            });
```

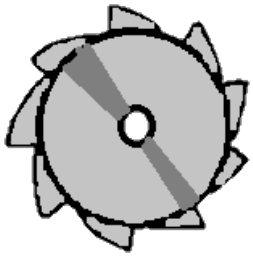
Na ovom programskom kodu potrebno je uočiti da se kontroliranje našeg objekta može dogoditi doticanjem i pomicanjem prsta na bilo kojem dijelu ekrana, no da sam objekt kojeg kontroliramo neće promijeniti vrijednost svoje x osi, odnosno neće se pomicati vertikalno, već se isključivo mijenja vrijednost položaja na horizontalnoj osi; također je bitno uočiti da za pomicanje objekta nije potrebno stisnuti na objekt te ga povući, već je dovoljno stisnuti na određeno mjesto na ekranu te će se objektov položaj na horizonalnoj osi teleportirati na željeno mjesto. Iako je ovo programsko riješenje vjerojatno najjednostavnije, kasnije će se ispostaviti da je ujedno i najbolje. Ukratko, može se dogoditi da objekt kojeg kontroliramo bude okružen s padajućim objektima koji se ne smiju pokupiti i stoga je ova mogućnost "teleportiranja" poželjna.

Kao što je već ranije navedeno, objekt za skupljanje koji se kontrolira ne mijenja svoj položaj na ekranu (bez prethodne akcije korisnika) već je potreba učitavanja više slika, odnosno stvaranja više Bitmap objekata nastala u svrhu stvaranja animacije. Animacija se stvara tako što se brzo izmijenjuju slike koje se iscrtavaju na zaslonu. Konkretno, ova animacija sastoji se od dvije slike koje se neprestano izmijenuju. Prva slika nastala je jednostavnim crtanjem, dok je druga nastala rotacijom za određeni kut te dodavanjem dodatnih detalja.

Sljedeće dvije slike prikazuju objekt s kojim skupljamo padajuće objekte, odnosno zajedno prikazuju oštricu koja se rotira velikom brzinom.



Slika 3.2: Originalna slika oštrice



Slika 3.3: Zarotirana originalna slika oštrice sa dodatnim detaljima

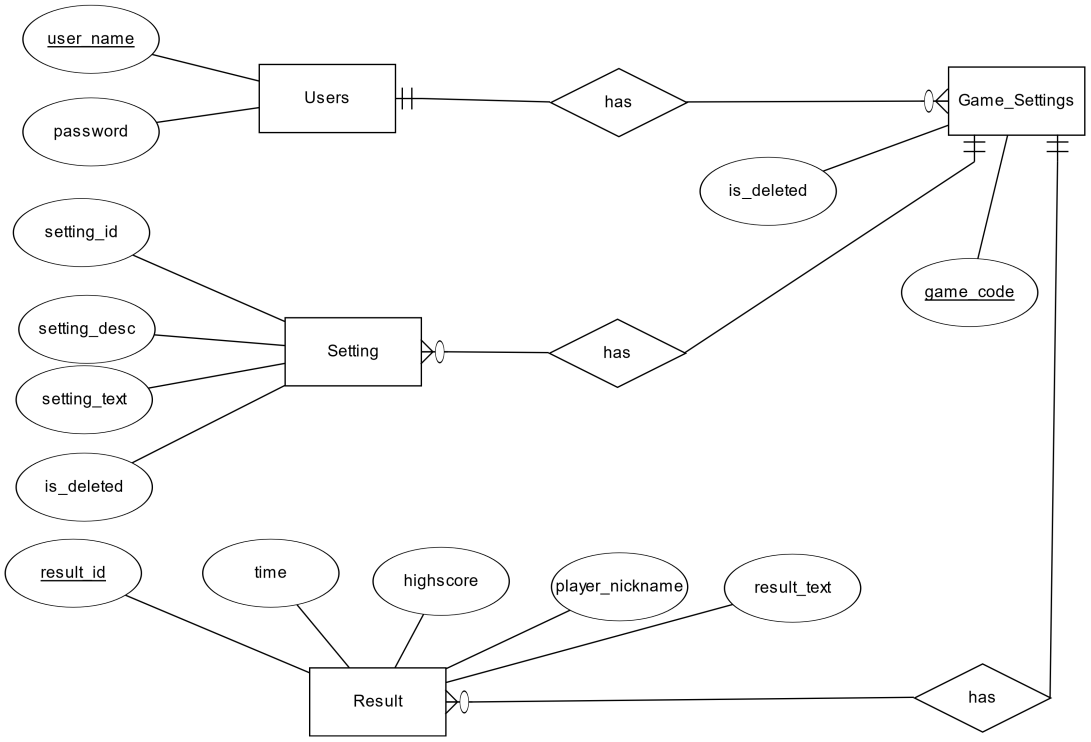
3.3. Padajući objekti

4. Baza podataka

Baze podataka su organizirane kolekcija podataka koje omogućavaju laki pristup podacima, uređivanje podataka i upravljanje istih. U sklopu ovog projekta koristio se SQL tip baze podataka, konkretnije PostgreSQL. PostgreSQL je besplatni, open-source sustav za upravljanje bazom podataka (SUBP) iza kojeg je više od 30 godina aktivnog razvoja.

4.1. ER dijagram

Sljedeća slika prikazuje ER dijagram baze podataka.



Slika 4.1: ER dijagram baze podataka

4.2. Opis entiteta i tablica

Users Ovaj entitet označava korisnika koji koristi web aplikaciju, odnosno web sučelje za učitelja. Sadrži atribute: user_ name i password. Atribut user_ name predstavlja korisničko ime korisnika, dok atribut password označava zaporku koja se ne sprema u "plain-textu", već kao rezultat hash funkcije. Ovaj entitet u vezi je *One-To-Many* s entitetom Game_ Settings preko atributa user_name.

Users		
user_name	VARCHAR	jedinstveni naziv korisnika
password	VARCHAR	rezultat hash funkcije nad zaporkom

Game_ Settings Ovaj entitet označava jedan "game room". Sadrži atribute: game_ code i is_ deleted. Ovaj entitet u vezi je *Many-To-One* s entitetom Users preko atributa user_ name, u vezi *One-To-Many* s entitetom Setting preko atributa game_ code te u vezi *One-To-Many* s entitetom Result preko atributa game_ code.

Game_ Settings		
game_ code	INT	jedinstveni idenfitikator game_ settings
is_ deleted	INT	zastavica koja govori jesu li "game room" obrisan
user_ name	VARCHAR	oznaka korisnika kojem "game room" pripada

Setting Ovaj entitet označava jedanu postavku. Sadrži atribute: setting_ id, setting_ desc, setting_ text i is_ deleted. Ovaj entitet u vezi je *Many-To-One* s entitetom Game_ Settings preko atributa game_ code.

Setting		
setting_ id	INT	jedinstveni idenfitikator za Setting
setting_desc	VARCHAR	tekst koji opisuje cilj zadatka
setting_text	VARCHAR	postavka formatirana tako da bude razumljiva mobilnoj igri
is_ deleted	INT	zastavica koja govori je li postavka obrisana
game_code	INT	oznaka Game_Settings kojem Setting pripada

Result Ovaj entitet označava jedan rezultat igranja igre na mobitelu uz preuzete postavke. Sadrži atribute: result_ id, time, highscore, player_ nickname te result_ text. Ovaj entitet u vezi je *Many-To-One* s entitetom Game_ Settings preko atributa game_ code.

Result		
result_ id	INT	jedinstveni idenfitikator za Result
time	VARCHAR	vrijeme pohrane rezultata u bazu podatka
highscore	VARCHAR	ostvareni rezultat na igri
player_ nickname	VARCHAR	nadimak igrača
result_text	VARCHAR	JSON polje s događajima igre
game_code	INT	oznaka Game_Settings kojem Result pripada

4.3. Relacijski dijagram

Sljedeća slika prikazuje relacijski dijagram baze podataka.

Slika 4.2: Relacijska shema baze podataka

5. Naslov 4

Poglavlje 4

Mobilna igra za vježbanje matematike

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Mobile game for practicing math

Abstract

Abstract.

Keywords: Keywords.