Izrada didaktičke igre u JavaScriptu

Pučić, Leo

Sveučilište u Rijeci – Odjel za informatiku
Jednopredmetna informatika

Leo Pučić

Izrada didaktičke igre u JavaScriptu

Završni rad

Mentor: Doc. Dr. Sc. Martina Holenko Dlab

Rijeka, rujan 2020.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 17. 6. 2020.

Zadatak za završni rad

Pristupnik: Leo Pučić

Naziv završnog rada: Izrada didaktičke igre u JavaScriptu

Naziv završnog rada na eng. jeziku: Creating a serious game in JavaScript

Sadržaj zadatka: Cilj primjene didaktičkih računalnih igara je omogućiti učenicima usvajanje nastavnog gradiva na zanimljiviji način pa uz elemente igre, ove aplikacije imaju i edukativnu komponentu. Zadatak završnog rada je opisati pojam didaktičkih računalnih igara i najčešće pristupe koji se koriste u njihovom razvoju. Potrebno je osmisliti vlastiti primjer didaktičke igre i opisati njenu svrhu, a zatim osmišljenu igru izraditi koristeći JavaScript te opisati postupak razvoja igre.

Mentor

Doc. dr. sc. Martina Holenko Dlab

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet:

Leo Pučić

(potpis pristupnika)

Sažetak

Igra je pojam koji je u današnjem društvu vrlo aktualan, bilo to društvena igra, kartaška igra ili u današnje doba računalne igre koje sve više dobivaju na popularnosti. Računalne igre su oblik razbibrige mnogima i služe za relaksaciju, a pritom imaju i edukativni karakter. Dijete u svakom trenutku uči i stvara svijest o naučenome, upravo didaktičke igre su te koje omogućuju djetetov intelektualni razvoj kroz igru na zanimljiv i zabavan način. Didaktičke igre imaju prvenstveno zadatak da utječu na intelektualni razvoj djeteta u čijoj osnovi je umni napor, koncentracija i sposobnost shvaćanja.

Cilj ovoga rada je na temelju analize i pregleda dostupne literature na zadanu temu analizirati važnost pozitivnih i negativnih učinaka igranja računalnih igara na djecu osnovne škole, te na temelju stečenog znanja izraditi vlastitu računalnu igru. Izrađena računalna igra je dio praktičnoga dijela ovog rada i zove se Qubova Avantura. Glavni je zadatak da učenici navigiraju labirintom te pronađu škrinje s pitanjem kako bi im se otvorila vrata koja ih vode na sljedeću razinu, a kako bi došli do tih škrinja moraju pronaći tri gumba koja otvaraju vrata do tih škrinja i postavi im se pitanje na koje je potrebno odgovoriti. Svaka razina ima tri pitanja, te su pitanja iz Hrvatskog jezika, Matematike i Prirode i društva.

Izrađena računalna igra može služiti korisnom učenju, razvoju brzine i razvoju koncentracije prilikom odgovaranja na zadana pitanja, te na samom kraju i zabavu kroz učenje što je djeci svakako i najzanimljivije.

Ključne riječi: *didaktičke računalne igre, učenici, učenje, motivacija, Qubova Avantura.*

Sadržaj

1. Uvod

Tema ovog završnog rada je *„Izrada didaktičke igre u JavaScriptu“*. Tema je vrlo aktualna, igre su sve više zastupljene među djecom školske dobi i dio su njihove svakodnevice te sam upravo iz tog razloga odlučio se za odabir ove teme.

Igra je glavna komponenta u djetetovom razvoju. Djeca od najranije dobi uče upravo kroz igranje, one im pružaju zadovoljstvo, motiviraju ih i u potpunosti su okupirani tom aktivnošću. Igranjem djeca zapravo uče nove vještine. Naučene vještine primjenjuju u drugim aktivnostima i grade vlastita iskustva. Potencijal računalnih igara u posljednjih nekoliko desetljeća potaklo je mnoge učitelje i edukatore da pokušaju što bolje i kvalitetnije implementirati računalne igre u nastavni proces. Zapravo, računalne igre služe kao izvrstan alat u učenju određenog predmeta ili za usvajanje novih znanja i vještina. Računale igre moraju prije svega biti prilagođene djetetovoj dobi i njegovim sposobnostima [1].

Cilj ovoga rada je na temelju provedene analize i pregleda dostupne literature na zadanu temu analizirati važnost pozitivnih i negativnih učinaka igranja računalnih igara na djecu za cijelo osnovno školstvo.

U prvom dijelu ovog završnog rada objašnjeni su pojmovi didaktike, didaktičkih igara i didaktičkih računalnih igara. Definirani su pojmovi igrifikacije i učenja kroz igru. Kronološki je poredano i opisano devet primjera didaktičkih računalnih igara. Na kraju prvog dijela ovog završnog rada objašnjena je važnost računalnih igara u nastavi.

U drugom dijelu ovog završnog rada ukratko je objašnjen programski jezik *JavaScript*, opisan je vlastiti primjer didaktičke računalne igre te su opisani glavni elementi aplikacije. Na kraju ovog rada opisane su mehanike igre i programski kod.

2. Didaktičke igre

Didaktičke igre imaju prvenstveno zadatak da utječu na intelektualni razvoj djeteta u čijoj osnovi je umni napor, koncentracija i sposobnost shvaćanja. Ove igre mogu biti s igračkama i raznim predmetima, stolne tiskane igre i igre riječima. Takvoj vrsti igre zadatak je da proširuje i bogati dječje znanje [2]. Djeca pomoću tih igara rješavaju zadatke iz svakidašnjeg života te slijede točno određeni tok igre kao i redoslijed te sustav same igre. Od djece zahtijevaju veću koncentraciju pažnje, sposobnost samosvladavanja, zahtijevaju od djeteta aktivno pamćenje sadržaja i pravila igre u svrhu ponavljanja i aktivacije intelektualne sposobnosti te samim time ulaže svoj napor što je svakako za dijete od velike važnosti.

Didaktičkom igrom se stoga smatra igra koja slijedi određena pravila u svrhu učenja. Glavna karakteristika igre je postizanje određenog rezultata, dok su kompetencije koje mogu biti stečene prilikom igranja ustrajnost, kritičko razmišljanje ili spremnost na rizik. Primjer takvih didaktičkih igara koje pomažu u jačanju kompetencije baziraju se na igrama lokacije i strategije [3].

Prednosti koje pružaju didaktičke igre u odnosu na e-obrazovanje je da imaju realistično okruženje koje omogućuje igračima jasno otkrivanje granica i saznavanje njihovih opcija a uz to imaju i jasno definirana pravila. Takve igre imaju definiranu svrhu i vrlo su interaktivne što bitno utječe na tijek same igre. Prate igračev napredak i pritom se prilagođavaju igraču kako mu ne bi dosadile ili frustrirale ga. Zahtijevaju od igrača da kognitivno sudjeluje u igri, odnosno tjeraju ga na kreativno razmišljanje, istraživanje i razmišljanje o posljedicama. Zbog tih posljedica igrač shvaća njihov ishod koji se bazira njegovim odlukama i ponašanjima. Jedna od najvažnijih prednosti didaktičkih igara je da igraču pruža zadovoljstvo tijekom njihovog igranja i upravo zbog toga igrač je duže koncentriran i ima veću pozornost [4].

2.1. Pojam didaktike

Didaktika je riječ grčkog podrijetla i ona u prijevodu znači poučavati, predavati, propisivati. Klasificira se kao grana pedagogije i bavi se prikupljanjem teorija, ideja, načela i uputa koji su usmjereni ka uspješnom provođenju odgojno-obrazovnih procesa [5].

Osnivači didaktike su njemački reformator edukacije Wolfgang Ratke i češki pedagog Jan Amos Komenský. Didaktiku zapravo smatramo kao izvorište ukupne

pedagogije, bavi se pitanjem svrhe i zadataka nastave koje vidi u obrazovnim i odgojnim ciljevima. U zadatke obrazovnih ciljeva ubraja se stjecanje znanja i razvijanje sposobnosti dok u zadacima odgojnih ciljeva ubrajaju se razvijanje interesa, afirmacija racionalnih stavova i poticanje korisnih potreba. Kao teorija o nastavi, didaktika istražuje djelotvornost same nastave tako da formulira didaktička načela kao svojevrsnu zakonitost koja se temelji na logičkoj vezi „kad-ako-onda“. Didaktika traži uspješne načine da se ostvare pojedini ciljevi te uz pomoć već provjerenih nastavnih metoda, pokušava što bolje prilagoditi nastavu veličini obrazovnih skupina, promišlja tijek nastave i oblikuje ga naspram logike spoznajnog procesa i samog procesa učenja. Također, primjenjuje i razvija sustav praćenja, ocjenjivanja i vrjednovanja učinaka nastave [6].

Analiza i planiranje ogdojno-obrazovnih procesa	**Osiguravanje praktičnih smjernica za djelovanje**	**Omogućuje uvid u didaktičke teorije i pravce**
• Istražuje nastavnu praksu putem različitih metoda • Razvija metodologiju odgojno-obrazovnih procesa	• Daje smjernice, upute koje valja shvatiti kao pomoć učitelju u poučavanju i učenju	• Različiti teorijski pristupi donose različita određenja didaktičkih ideja i različite teorijske modele koji se međusobno nadopunjavaju, a koji se mogu propitkivati i mijenjati u praksi

Tablica 1. Tri glavna zadatka didaktike

2.2. Učenje kroz igru i igrifikacija

Veća uporaba didaktičkih igara počela je predstavljati problem u razlikovanju pojmova kao što je učenje kroz igru (*game-based learning*) i igrifikacija (*gamification*). Ova dva navedena pojma su bitno različita i u narednim poglavljima objašnjeno je koje su njihove razlike.

2.2.1. Učenje kroz igru (Game-Based Learning)

Učenje koje koristi elemente igara i primjenjuje ga na usvajanje novih vještina ili postizanje određenog ishoda učenja. Odnosno, uzima osnovni sadržaj i ciljeve te ga čini zabavnim. Ukomponiranje takvog učenja omogućava korisnicima da nauče nove

koncepte i uvježbavaju vještine bez ikakvog rizika. Njihov napredak u igri je direktno povezan sa shvaćanjem predmeta kojeg uče. Korisnici su više motivirani takvim učenjem jer se istodobno i zabavljaju.

Nedostatak učenja kroz igru je taj jer je potrebno graditi nove module učenja, iziskuje puno vremena na kreiranje i skupo je. Ovaj nedostatak može bitno utjecati na njegov odabir [7].

2.2.2. Igrifikacija

Igrifikacija je korištenje mehanika igre i primjenjuje ih u aktivnostima koje nisu vezane uz igru kako bi se postiglo željeno ponašanje i kako bi se potaklo učenje. U posljednjih nekoliko godina naglim razvojem mobilnih uređaja potaknulo je mnoge programere i informatičke tvrtke da u svojim aplikacijama ukomponiraju igrifikaciju [7]. Tablica 2 prikazuje par notabilnih mehanika igre koje se često koriste u igrificiranju mobilnih aplikacija i ukratko su objašnjene.

Značke postignuća	• vizualni pregled korisničkih postignuća u igri
Razine	• dio igre, što je veća razina to je i težina igre veća
Dijagram performansi	• prikazuje korisnikov trenutni rezultat i uspoređuje ga sa prijašnjim rezultatom
Bodovi	• nagrada koju korisnik dobiva rješavajući jednostavne zadatke
Tablica rezultata	• tablični prikaz korisnikovih performansi u odnosu na druge korisnike
Valuta	• najčešće novi život ili ponovni pokušaj kojeg korisnik dobiva nakon obaljanja nekog zadatka

Tablica 2. Mehanike igre koje se koriste u igrifikaciji

Takve elemente je moguće implementirati samo specijaliziranim softverom koji je namijenjen za implementaciju igrifikacije. Među najpoznatijim softverima za implementiranje igrifikacije u mobilnim aplikacijama ubrajamo: *Amazon GameCircle*, *Apple's Game Center* i *Google Game Services* [8].

Prednost igrifikacije je da nije potrebno kreirati novi sadržaj, igrificiranje brz je i jednostavan proces a ono najvažnije je da je jeftina za implementiranje. Također motivira korisnike i pozitivno utječe na njihovo učenje. Tjeraju se da budu što bolji od drugih, da postignu visok rezultat na ljestvici i osvoje mnogo nagrada i bodova.

Implementacija igrifikacije nije uvijek najbolja strategija za sve tipove učenja. Najbolje je implementirana u sadržaju koji je jednostavan za memoriranje i nije kompleksan korisnicima za korištenje. Pošto učenje nije igrificirano teže je ostati motiviranim i sadržaj učenja mora biti jednostavan za razumjeti. Ako korisniku treba dugo vremena provoditi učeći, motivacija se gubi i efektivnost igrifikacije se gubi [7].

Igrifikacija nastavnog sadržaja zapravo je jedan od boljih načina kako pristupiti u razvoju računalnih igara jer omogućuje programerima da naprave što bolju i kvalitetniju igru.

2.3. Didaktičke računalne igre

Didaktičke računalne igre počinju sve više dobivati na popularnosti jer na zabavan i interaktivan način zaintrigiraju učenike i motiviraju ih na usavršavanje i ponavljanje određenog sadržaja. Skica 1 prikazuje najvažnije karakteristike didaktičkih računalnih igara. Iz ove skice zaključujemo da didaktičke računalne igre imaju definirana pravila i određeni cilj. Predstavljaju izazov korisnicima i interaktivne su. Potiču suradnju s drugima i učvršćuju socijalne vještine korisnika. Stimuliraju osjetila prilikom ulaska u virtualne svjetove i potiču korisnika na bujnu maštu. Dobivaju povratne informacije koje ga usmjeravaju ka boljem postizanju cilja i rješavanju zadataka [9].

Skica 1. Karakteristike didaktičkih računalnih igara

Edutainment je vrsta računalnog softvera, a to je kombinacija računalnih igara i edukativnog softvera. U prvom planu je da se dijete zabavi tijekom igranja, a u drugom planu je da nešto i nauči prilikom igranja. Takva vrsta igre se ne ukomponira u nastavni proces nego je predviđena za igranje kod kuće. Najčešće su to igre s poznatim likovima

iz crtića i vrlo vjerojatno će zaintrigirati dijete da ju pokrene i počne igrati. U trgovinama se često obilježavaju kao vrsta edukativnog softvera bez obzira na to što nemaju neku edukativnu svrhu [10].

Didaktičke računalne igre ne moraju biti edukativnog sadržaja da bi igrača naučile nekim vještinama. Takve igre koje nemaju edukativnu svrhu, a svejedno ponešto nauče igrača, zovemo ozbiljnim igrama (eng. *serious games*). Postoji puno kvalitetnih ozbiljnih igara i u Tablici 3 je prikazano nekoliko dobrih primjera te koje edukativne elemente sadrže [11].

Age of Mythology	•Strategija, mitologija i povijest
Game Dev Tycoon	•Poslovno planiranje i mendžment
LittleBigPlanet	•Dizajn, kreativnost i prostorno razmišljanje
Monkey Island	•Rješavanje zagonetki, istraživanje
Plant Tycoon	•Fitologija, planiranje
Scribblenauts	•Učenje stranog jezika, razmišljanje
SimCity 4	•Geografija, Urbanizam
Spore	•Evolucija, genetika, astronomija
The Sims	•Čovjekova zanimanja, arhitektura
Zoo Tycoon 2	•Zoologija

Tablica 3. Prikaz poznatih računalnih igara i njihove edukativne elemente

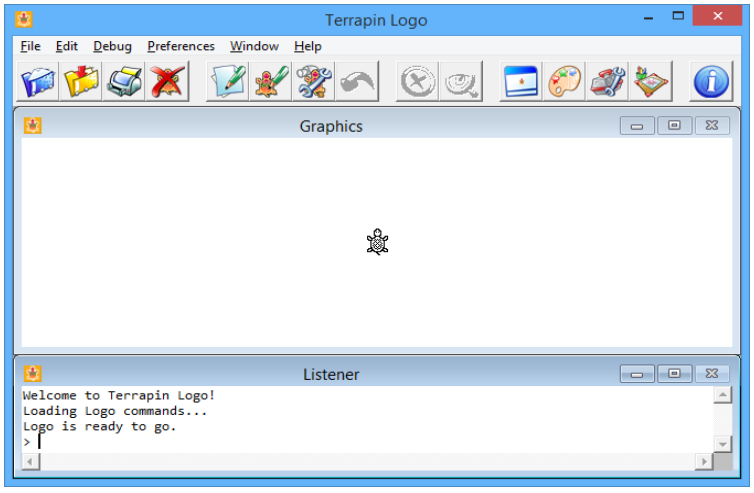
Iz ove tablice vidljivo je da većina navedenih računalnih igara je tipa simulacije. Takav tip igre motivira i potiče igrača na logičko razmišljanje, strategiranje, prostorno planiranje i jača njegove sposobnosti za kritičko razmišljanje, rješavanje zagonetki i prostornog planiranja.

2.4. Povijest didaktičkih računalnih igara

Naglim razvojem osobnih računala 80-ih godina pridonijelo se i naglom razvoju računalnih igara. Primijetio se ogromni potencijal računalnih igara i igraće industrije. Tako se rodila ideja da se u računalnim igrama implementiraju edukacijski elementi te da se počnu koristiti u nastavnom procesu kao dodatni alat u učenju. U sljedećim poglavljima navedeno je devet primjera didaktičkih računalnih igara poredanih u kronološkom redu i ukratko su opisane.

2.4.1. Programski jezik Logo (1967)

Među prvim pokušajima ukomponiranja edukacijskih elemenata u računalne igre smatra se programski jezik *Logo*. Osmislili su ga američki matematičar Seymour Papert i računalni znanstvenik Wally Feurzeig 1967. godine. *Logo* je jezik opće namjene. Svrha ovog programskog jezika je da učenike nauči osnovama programiranja [12]. Koristi *Turtle* grafiku i omogućuje da se jednostavnim naredbama pokreću kornjačom ili pokazivačem i crtaju linije i vektore na ekranu [13]. Postoji više varijanti programskog jezika *Logo*. A među poznatijim varijantama ubrajaju se: *Lego Logo*, *ObjectLOGO* i *Atari Logo* [14] [15] [16]. Na Slici 1 prikazana je novija verzija programa *Terrapin Logo* koji je među popularnijim inačicama *Logo-a* koji se i danas koristi u nastavnom procesu predmeta informatike u osnovnim školama.



Slika 1. Grafičko sučelje novije verzije programa Terrapin Logo

2.4.2. The Oregon Trail (1971)

The Oregon Trail su razvili učitelji Povijesti Don Rawitsch, Bill Heinemann i Paul Dillenberger 1971. godine, a tvrtka *MECC* objavljuje igru 1974. godine. Početna verzija je bila tekstualnog oblika no daljnjim razvojem dodano je i grafičko sučelje i puno novih značajki. Postala je jedna od najpopularnijih edukativnih računalnih igara i često se mogla naći u informatičkim učionicama ili školskim knjižnicama tijekom 70-ih i 80-ih godina. Učenici su u pionirskim danima 1800-ih učili voziti volove iz Missourija do Oregona. Na putovanju su nailazili na prepreke i izvršavali zadatke koji su bili „realni“ za to vremensko razdoblje. Tako je igra *The Oregon Trail* integrirala lekciju iz američke povijesti u interaktivno iskustvo [17]. Obilježila je mnoga djetinjstva i ušla u povijest kao jedna od najboljih edukativnih računalnih igara svih

vremena. Iznjedrilo je mnogo nastavka, mnogi su pokušali kopirati i poboljšati koncept ove igre, ali malo tko je uspio.

2.4.3. Math Blaster! (1983)

Igru je osmislio edukator Jan Davidson 1983. godine. To je edukativna računalna igra u arkadnom stilu koja nudi matematičke vježbe kako bi pružila učenicima da nauče i ponavljaju gradivo iz Matematike. Namijenjena je za korištenje od prvog do šestog razreda osnovne škole. Aktivnosti učenja bile su grafički privlačne i djelovale su pozitivno na djetetovu motivaciju za boljim učenjem. Daljnjim testiranjima u osnovnim školama igra se razvijala i poboljšavala kako bi novije verzije *Math Blaster*! serijala bile što bolje i kvalitetnije. Prodala se u mnogo primjeraka i ostvarila status najboljeg matematičkog softvera [18].

2.4.4. Where in the World is Carmen Sanadiego (1985)

Edukativna računalna igra koju je objavila tvrtka *Broderbund* 1985. godine. To je prva igra u serijalu *Carmen Sandiego*. Igrač preuzima ulogu novaka u izmišljenoj detektivskoj agenciji *ACME*, čiji je zadatak ući u trag prevarantima iz *V.I.L.E.* organizacije koja je ukrala poznata djela iz cijelog svijeta. Igrači, koristeći svoje znanje iz Geografije, ispituju svjedoke ili istražuju tragove kako bi pronašli lopova. Uspješnim rješavanjem ovih zločina povećava se igračev rang, zadaci tako postaju sve teži i na kraju dobivaju zadatak da pronađu vođu *V.I.L.E.* organizacije i glavnu junakinju Carmen Sandiego [19]. Fantastičan primjer igre koji ukomponira posao iz stvarnog života i primjenjuje ga u izmišljenoj radnji koja će sigurno zaokupirati igrača i potaknuti ga na kontinuirano igranje, a pritom će ponešto i naučiti iz geografije ili povijesti, a da nije svjestan da zapravo usvaja ili učvršćuje nove vještine.

2.4.5. Elmo`s Preschool (1996)

Zabavna igra namijenjena djeci predškolske dobi. *Creative Wonders* i *Children`s Television Workshop* razvile su i objavile igru 1996. godine. Poznati likovi iz Ulice Sezam na interaktivan i zabavan način uče djecu o brojanju, čitanju abecede, raspoznavanju geometrijskih likova, upoznaju ih s glazbenim instrumentima i uče ih o ljudskim emocijama [20]. Igra je grafički interesantna i privlači djetetovu pažnju, žarke boje i zabavni likovi će sigurno izazvati dječju radost i polako ih pripremiti za osnovnu školu.

2.4.6. CD Učilica (2002)

CD Učilica je interaktivna računalna igra s edukativnim elementima koja je namijenjena djeci od prvog do osmog razreda osnovnih škola. Služi kao dodatni alat u učenju i ponavljanju školskog gradiva na zabavan način. Raspolaže s četrdeset tisuća pitanja koja su pažljivo odabrana i prilagođena trenutnom nastavnom programu. Osim učenja i ponavljanja školskog gradiva *CD* Učilica ima pregršt zanimljivih i zabavnih aktivnosti kao što je učenje prometnih pravila i prometnih znakova, učenje brzog tipkanja, učenje stranih jezika i učenje o športu, tjelesnim aktivnostima i o važnostima zdrave prehrane [21]. Mnogim učenicima je korištenje *CD* Učilice pomoglo u boljem savladavanju gradiva i poboljšanju ocjena u školi.

2.4.7. National Geographic Challenge (2010)

The National Geaographic Challenge je interaktivni kviz namijenjen igraćim konzolama kao što su *PlayStation 3*, *Nintendo Wii* i *Xbox 360*. Objavljena je 2010. godine i razvila ju je tvrtka *UTV Ignition Games* u suradnji s *National Geographic-om*. Igra na zabavan i interaktivan način uči djecu i mlade o povijesnim figurama, povijesnim događajima i Geografiji. Raspolaže s pet tisuća različitih pitanja koja će mlade igrače izazvati i potaknuti na razmišljanje [22]. Savršen primjer igre za igranje u društvu. Igra ne samo da potiče na razmišljanje i usvajanje novih vještina, nego potiče socijalnu interakciju s ostalim članovima društva te jača čovjekove sociološke vještine, a sve se to odvija u zabavi.

2.4.8. World Rescue (2014)

World Rescue je pripovjedačka edukativna igra koju je razvila i objavila tvrtka *ZU Digital* u suradnji s *UNESCO MGIEP-om* 2014. godine. Igrač upoznaje petoricu heroja i pomaže im u rješavanju globalnih problema kao što su: nestašica hrane, poplave, krčenje šuma, suše, raznih bolesti, recikliranje otpada i još mnogo toga. Idealna je za učenje Geografije, Ekologije te potiče igrača na rješavanje različitih problema i tjera ga na logičko razmišljanje. Igra je dostupna za mobilne uređaje [23]. Igra je perfektan izbor za educiranje djece i mladih o globalnom zatopljenju i o posljedicama koje može uzrokovati i tako ih motivirati da se brinu o okolišu te zašto je važno da čuvamo naš planet.

2.4.9. Prodigy Math Game (2018)

Prodigy Math Game je edukativna mobilna igra namijenjena djeci od šest do dvanaest godina s ciljem da nauče i usavrše gradivo matematike. Igra pruža jedinstveno iskustvo učenja kroz interaktivno i zabavno učenje matematike u kojoj uspjeh ovisi o pravilnom odgovaranju na matematička pitanja. Igrači istražuju i rješavaju matematičke zagonetke, natječu se s drugim igračima i dobivaju razne nagrade te istodobno uče nove vještine. Igra je osvojila mnogo nagrada i postala je najboljom edukativnom mobilnom aplikacijom 2018. godine [24]. Djecu mlađe dobi priprema, na zabavan i jednostavan način, da usvoje osnove matematike i tako budu spremniji i upoznati s konceptom predmeta matematike. Osnovnoškolcima je izvrstan alat za učenje i ponavljanje gradiva matematike.

2.5. Važnost računalnih igara u školskom obrazovanju

Prema autorici Merrilea Mayo koja kaže da ima smisla da se računalne igre koriste kao alat u učenju. Smatra da računalna igra ima puno više prednosti nego tradicionalni školski sat. Navodi sljedeće prednosti:

1. Kompleksne igre se „razbijaju“ u dijelove i tako igrača vode korak po korak
2. Učenici imaju kontrolu kako će navigirati kroz igru
3. Igre kontinuirano i brzo daju povratne informacije učenicima
4. Igre se mogu prilagoditi učenikovom tempu
5. Zadaci temeljeni na igrama mogu zahtijevati od učenika da formiraju hipoteze i eksperimentiraju.

Svaka od ovih karakteristika je povezana s boljim ishodima učenja. Također, autor James Paul Gee navodi da popularnost teških i kompleksnih računalnih igara pokazuje da su programeri postigli ono što mnogi tradicionalni pristupi obrazovanja nisu uspjeli. Kaže da su pogodili „izuzetno dobre metode kako ljude navesti da uče i uživaju u učenju“.

Iz toga zaključujemo da ukomponiranje računalnih igara u školsko obrazovanje je vrlo dobar način da učenici kroz proces igranja računalnih igara postanu motivirani i aktivni u usvajanju što više novih vještina jer im takav oblik učenja pruža zadovoljstvo, koncentrirani su i nisu pod stresom. Učenici pokušavaju pronaći što bolje načine kako pristupiti problemu te ga brzo i efikasno riješiti. No, treba paziti da se odabere adekvatna računalna igra koja će učenika motivirati da nauči pravu lekciju. Ako računalna igra nauči učenika krivu lekciju, onda ona kao takva nije adekvatna za

primjenu u školskom obrazovanju i treba izbjegavati korištenje takvih računalnih igara kako ne bi loše utjecale na učenikovo psihičko zdravlje [25].

3. JavaScript

JavaScript je programski jezik koji se koristi u razvoju raznih web aplikacija i smatra se kao jedan od najpopularnijih programskih jezika. Vrlo je jednostavan za korištenje i brzo ga se nauči [26].

Koristi se uz *HTML5* i *CSS* programske jezike za razvoj web aplikacija. Omogućuje interaktivne web stranice i važan he dio web aplikacija. Izvršava se na strani korisnika i mnogo web preglednika imaju poseban *JavaScript* mehanizam (eng. *engine*) za njegovo izvršavanje. Programske stilove koje podržava su funkcionalni, imperativni i stilovi koji su upravljani nekim događajima (eng. *event-driven*) [27]. Za *JavaScript* kažemo da je doslovno programski jezik web-a jer je postao standardnim programskim jezikom za korištenje i razvoj web-a i web aplikacija. Na Slici 2 je prikazan logo *JavaScript* programskog jezika.

Slika 2. Prikaz logo-a

3.1. Povijest JavaScripta

Nastao je u doba kada se Internet počeo naglo razvijati. *Netscape* je tada bio najpopularniji web preglednik i Brendan Eich, koji je tada radio za *Netscape Communication Corporation*, je u samo deset dana napravio programski jezik *JavaScript*, u devetom mjesecu, 1995. godine [28]. Na Slici 3 je prikazan zaslon *Netscape* web preglednika.

Slika 3. Prikaz zaslona Netscape web preglednika

Brendan Eich prilikom razvoja *JavaScript-a* bio je inspiriran *Java* programskim jezikom. Ljudi često brkaju ta dva programska jezika, a oni su bitno različiti i uopće nemaju sličnosti. U Tablici 4 prikazane su razlike između *Jave* i *JavaScript-a* [29].

Java	***JavaScript***
•Kompajliran •Koristi se u *back-endu* •Osigurava bolju sigurnost •Sintaksa je u programskom jeziku *C++* •Može se samo koristiti u *Java Development Kit-u* •Pogodna je za različite aplikacije	•Interpretiran •Koristi se u *front-endu* •Treba više uložiti truda kako bi se postigla bolja sigurnost •Sintaksa je slična programskom jeziku C •Može se razvijati u bilo kojem *editoru* •Koristi se najviše za kreiranje web aplikacija

Tablica 4. Prikazuje glavne razlike Java i JavaScript programskih jezika

3.2. Prednosti i ograničenja JavaScript programskog jezika

JavaScript programski jezik traži manju potrebu za komunikacijom sa serverom tako što provjerava ispravnost podataka prije slanja stranice na server i tako smanjuje promet prema serveru. Trenutni odziv korisniku, odnosno nije mu potrebno čekati da se stranica ponovno učita da vidi ako je zaboravio unijeti neki podatak ili ga krivo upisao. Povećava interaktivnost tako što ima mogućnost kreirati sučelje koje

reagira na korisnikovo korištenje tipkovnice i miša. Bogatije sučelje, naprimjer možemo koristiti *JavaScript* za animiranje bogatijih sučelja prema korisniku.

JavaScript kao i mnogi drugi programski jezici ima svoja ograničenja. Ne smatramo ga potpunim programskim jezikom jer mu nedostaje par važnih osobina. *JavaScript* je klijentski programski jezik i zbog sigurnosnih razloga nije mu dopušteno čitanje i pisanje datoteka. Ne možemo koristiti ga za mrežne aplikacije jer ne postoji podrška. Limitiran je tako što nema mogućnost višenitnog i višeprocesorskog izvođenja [30].

3.3. Kreiranje igara programskim jezikom JavaScript

JavaScript kao i mnogi drugi programski jezici može isto biti koristan alat za kreiranje i razvoj računalnih igara. Moguće je koristiti 2d i 3d programske biblioteke u kombinaciji s *JavaScript-om* za kreiranje računalnih igara u web pregledniku ili kao vanjske platforme mehanizma igre. Neki od alata koji se koriste u kombinaciji s *JavaScript-om* su: *HTML5, Canavas*, *WebCL*, *Phaser*, *Impact.js*, *HaXe*, *MelonJS*, *Threejs*. Prednost u korištenju ovih alata je da cijena razvijanja igre je minimalna te se stoga dodijeljeni resursi mogu znatno smanjiti [31].

Primjeri dobrih igara koje je moguće igrati preko poznatih web preglednika, a da koriste *JavaScript* u kombinaciji s drugim web alatima.

1. *HexGL* – brza trkaća igra koja je smještena u dalekoj budućnosti, sagrađena je korištenjem *HTML5*, *JavaScript* i *WebGL* alata
2. *Bejeweld* – klasična igra slaganja i premještanja dijamanata, izgrađena je pomoću *HTML5* i *JavaScript* alata
3. *Angry Birds* – popularna mobilna igra konvergirana je pomoću *HTML5* i *JavaScript* alata te ju je moguće igrati i preko web preglednika
4. *Swoop* – igra u kojoj se leti i skuplja dijamante, koristi alate za izradu lijepog i šarolikog 3D svijeta [32].

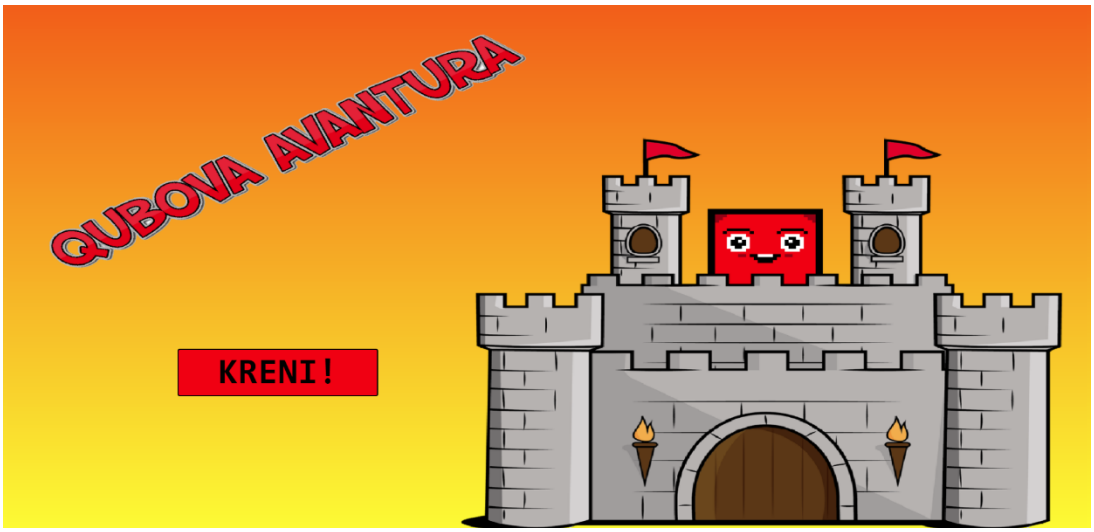
4. Qubova Avantura

Izrađena didaktička računalna igra namijenjena je u edukativne svrhe za djecu od prvog do četvrtog razreda osnovne škole, stoga je i sam izgled kao i navigacija stranice vrlo jednostavnog karaktera, ali dovoljno da zaintrigira učenika na igranje.

Cilj izrađene igre je da učenici od prvog do četvrtog razreda ponavljaju i uče gradivo Hrvatskog jezika, Matematika i Prirode i društva, a pritom da nisu svjesni da zapravo uče i ponavljaju gradivo. Učenicima je prije početka igre predstavljena zanimljiva priča koja ih motivira da pomognu „*Qubo-u*“ i motivirani su na rješavanje zadatka koji im je postavljen. Navigiraju labirintom i razmišljaju koji put je najbolje uzeti kako bi došli do izlaza koji ih vodi na sljedeću razinu. Prepreke koje ih sprječavaju i izazivaju u rješavanju tog zadatka su upravo te da moraju pronaći pravi gumb koji otključava vrata odgovarajuće boje i tako im se oslobodi put do škrinje u kojoj se nalazi pitanje. Kada otvore sve škrinje s pitanjima i odgovore na sva pitanja, otvaraju se posljednja vrata koja su crvene boje. Oslobađa im se put ka izlazu i omogućen im je prelazak na sljedeću razinu. Ukupno je tri razine i nakon prelaska posljednje razine učenik dobiva poruku s ukupnim brojem bodova. Pitanja su prilagođena djeci od prvog do četvrtog razreda osnovne škole i baziraju se na nastavnom planu i programu za osnovne škole. Jednostavnog su oblika i učeniku ne predstavlja problem u čitanju. Svako pitanje ima ponuđena tri odgovora od kojih je samo jedan točan. Učenika tako igra motivira na ponovno igranje kako bi postigao što bolji rezultat, a uz to i da ponovi naučeno gradivo.

4.1. Početna stranica

Primjer skice početne stranice vidljiv je iz daljnje slike na kojoj je prikazan naslov igre „*Qubova* Avantura“, gumb „KRENI!“ i jednostavna ilustracija s glavnim likom. Klikom na gumb „KRENI!“ prelazi se na zaslon gdje „*Qubo*“ priča što mu se dogodilo i traži igračevu pomoć, a to je sve vidljivo sa Slike 5.



Slika 4. Prikaz naslovne stranice

Slika 5. Prikaz Qubove priče

4.2. Stranica pravila igre

Na ovoj stranici prikazano je kako se kontrolira „*Quba*“ i ciljeve koje igrač mora ispuniti tijekom igranja. U donjem desnom kutu nalazi se gumb „IGRAJ!“ koji dovodi igrača do stranice za odabir razreda. Izgled stranice pravila igre prikazan je na Slici 6.

Slika 6. Prikaz pravila igre

4.3. Stranica za odabir razreda

Glavni dio stranice za odabir razreda podijeljen je u pet dijelova te se sastoji od odabira prva četiri razreda osnovne škole i na kraju se nalazi gumb „Početna stranica“ za povratak na naslovnu stranicu. Izgled stranice za odabir razreda možemo vidjeti na Slici 5.

Slika 7. Prikaz zaslona za odabir razreda

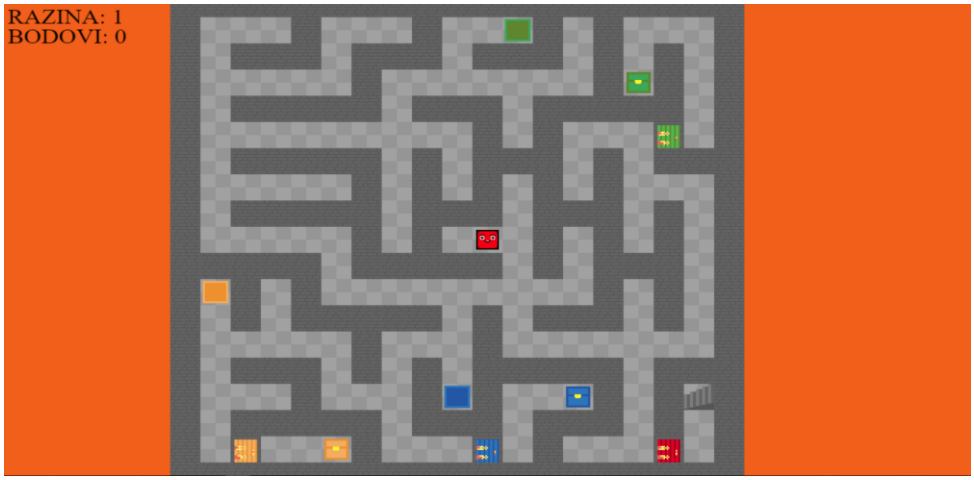
4.4. Stranica za odabir predmeta

Nakon što smo odabrali željeni razred, imamo tri gumba za odabir predmeta. Predmete koje je moguće odabrati jesu Hrvatski jezik, Matematika i Priroda i društvo te gumb „Odabir razreda“ za povratak na stranicu za odabir razreda. Izgled stranice za odabir predmeta vidljiv je na Slici 8.

Slika 8. Prikaz zaslona za odabir predmeta

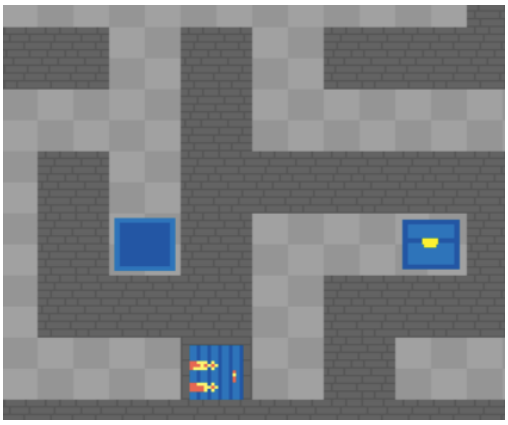
4.5. Prikaz igre

Nakon što odaberemo željeni razred i predmet otvara nam se zaslon u kojem se nalazi prikaz igre, a to možemo vidjeti na Slici 9.



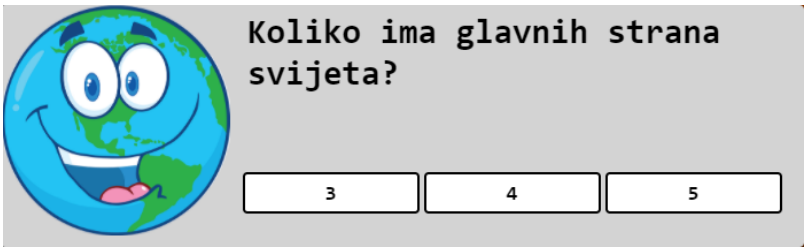
Slika 9. Prikaz zaslona igre Qubova avantura

Igra se sastoji od labirinta u kojem se nalaze gumbi koji otvaraju vrata i škrinje s pitanjima. U lijevom gornjem kutu možemo vidjeti na kojoj smo razini i trenutno stanje bodova. Zeleni gumb otvara zelena vrata koja oslobađaju put do zelene škrinje s pitanjem. Isto to vrijedi za narančastu i plavu boju tih objekata. Na Slici 10 prikazani su svi objekti plave boje.



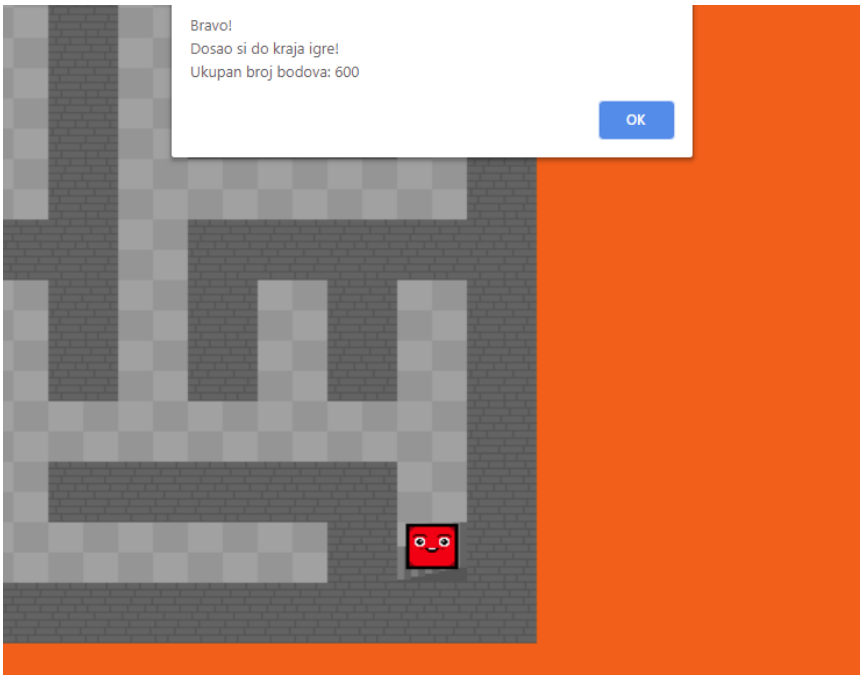
Slika 10. Prikaz zaslona igre u kojem se nalaze objekti

Kada učenik stane na škrinju, otvara mu se pitanje s tri ponuđena odgovora. Odgovori li točno na postavljeno pitanje, dobiva sto bodova. Točan odgovor je popraćen adekvatnim zvukom koji učeniku daje na znanje da je točno odgovorio na pitanje. Odgovori li učenik krivo oduzima mu se deset bodova i dobiva novo pitanje sve dok ne odgovori točno. Krivi odgovor je također popraćen adekvatnim zvukom. Primjer prozora s pitanjem prikazan je na Slici 11.



Slika 11. Prikaz zaslona s primjerom pitanja

Nakon što učenik pronađe sve gumbe i otvori vrata koja vode do škrinje s pitanjem i odgovori na postavljena pitanja, zvučnim signalom se otvaraju posljednja vrata koja su crvene boje i koja vode do stepenica za prelazak na sljedeću razinu. Ukupno je tri razine i na svakoj razini su po tri škrinje s pitanjem koje učenik mora odgovoriti kako bi došao do kraja igre i oslobodio *Quba* iz začaranog labirinta. Kada učenik dođe do kraja igre na ekranu mu se prikaže ukupan broj bodova i vraća ga na naslovnu stranicu. Slika 12 prikazuje poruku s ukupnim brojem bodova.



Slika 12. Prikaz zaslona s porukom nakon završetka igre

5. Mehanika igre Qubova Avantura

Igra *Qubova* Avantura je kreirana primarno korištenjem *JavaScript* programskog jezika. *HTML5* i *CSS* su korišteni samo za izradu web stranice, odnosno njenog izgleda i stila. U sljedećim poglavljima objašnjena je izrada mehanika kretanja i vidnog polja, zatim mehanika koja crta mapu i ostale objekte koji su vidljivi na zaslonu. Također, implementiran je modal, vrsta dijaloškog okvira. Aktivira se kada se izvrši neka radnja. Osim mehanike dijaloških okvira, opisane su i preostale funkcije i varijable.

5.1. Mehanike kretanja i vidnog polja igre

Da bi se lik mogao kretati, na mapi treba kreirati objekt naziva *Character*, slušatelje eventa i slušatelje klika, dodati proces kretanja i crtanje lika. Prvo što treba je dodati mapu kodova (eng. *keyCodes*) i zastavice (eng. *flags*) kojima se provjerava koja je tipka pritisnuta. Kodovi 37 do 40 predstavljaju tipke „lijevo“, „gore“, „desno“ i „dolje“, tim redoslijedom. Pretpostavit ćemo da nije pritisnuta niti jedna tipka i postavit ćemo zastavice na false. To se može vidjeti sa Slike 13.

```
var keysDown = {
    37 : false,
    38 : false,
    39 : false,
    40 : false
};
```

Slika 13. Varijabla keyDown

Zatim dodajemo još jednu globalnu varijablu koju nazovemo *player* i u nju spremamo novi objekt *Character*. Vidljivo na Slici 14.

```
var player = new Character();
```

Slika 14. Varijabla player

Kreiramo funkciju *Character* u kojoj spremamo podatke koordinata u kojima lik ide iz jedne pločice u drugu pločicu (*tileFrom*, *tileTo*), vrijeme u milisekundama koliko je proteklo od kad se počeo kretati (*timeMoved*), dimenzije lika (*dimensions*), trenutna pozicija u kojoj se lik nalazi na mapi i vrijeme u milisekundama koje će biti potrebno liku da se pomakne za jednu pločicu (*delayMove*). Sljedeći kod je prikazan na Slici 15.

```
//QUBO
function Character()
{
    //QUBOVA POZICIJA
    this.tileFrom   = [1,1];
    this.tileTo     = [1,1];
    this.timeMoved  = 0;
    this.dimensions = [30,30];
    this.position   = [45,45];
    this.delayMove  = 150; //QUBOVA BRZINA
```

Slika 15. Funkcija Character

Zatim našem liku (*Character*) pridružujemo metodu *placeAt* koja omogućuje da postavimo lika na bilo koju pločicu na mapi, što je vidljivo na Slici 16.

```
//SMJESTI QUBO-A
Character.prototype.placeAt = function(x, y)
{
    this.tileFrom   = [x,y];
    this.tileTo     = [x,y];
    this.position   = [((tileW*x)+((tileW-this.dimensions[0])/2)),
        ((tileH*y)+((tileH-this.dimensions[1])/2))];
};
```

Slika 16. Funkcija Character

```
//QUBO-OVO KRETANJE
Character.prototype.processMovement = function(t)
{
    if(this.tileFrom[0]===this.tileTo[0] && this.tileFrom[1]===this.tileTo[1]) { return false; }

    if((t-this.timeMoved)>=this.delayMove)
    {
        this.placeAt(this.tileTo[0], this.tileTo[1]);
```

Slika 17. Procesiranje kretanja 1. dio

Iz Slike 17 provjerava se je li vrijeme koje je prošlo otkako je lik započeo svoje kretanje jednako ili duže od vremena koje je potrebno liku da se pomakne za jednu pločicu. Ako je tvrdnja istinita onda postavljamo lika na odredišnu pločicu metodom *placeAt*. Provjerava se da li se lik još kreće, ako je istina, mora se točno izračunati njegov trenutni položaj na ploči odnosno labirintu.

```
    else
    {
        this.position[0] = (this.tileFrom[0] * tileW) + ((tileW-this.dimensions[0])/2);
        this.position[1] = (this.tileFrom[1] * tileH) + ((tileH-this.dimensions[1])/2);

        if(this.tileTo[0] !== this.tileFrom[0])
        {
            var diff = (tileW / this.delayMove) * (t-this.timeMoved);
            this.position[0]+= (this.tileTo[0]<this.tileFrom[0] ? 0 - diff : diff);
        }

        if(this.tileTo[1] !== this.tileFrom[1])
        {
            var diff = (tileH / this.delayMove) * (t-this.timeMoved);
            this.position[1]+= (this.tileTo[1]<this.tileFrom[1] ? 0 - diff : diff);
        }

        this.position[0] = Math.round(this.position[0]);
        this.position[1] = Math.round(this.position[1]);
    }
    return true;
}
```

Slika 18. Procesiranje kretanja 2. dio

Zatim iz Slike 18 provjeravamo da li se lik kreće vodoravno (os x). Ako je zaista tako, onda se izračunava broj piksela koliko se lik pomaknuo. Tada dijelimo širinu pločice s vremenom koje je potrebno da se lik pomakne za jedan korak i taj se rezultat množi s vremenom koje je poteklo od početka kretanja lika. Ista provjera se radi i za horizontalno kretanje. Nakon što se ažurira pozicija, zaokruže se vrijednost x i y na najbliži cijeli broj. Zatvara se funkcija i vraća se istinita vrijednost i daje do znanja programu da se lik još kreće.

```
window.addEventListener("keydown", function(e) {
    if(e.keyCode>=37 && e.keyCode<=40) { keysDown[e.keyCode] = true; }
```

Slika 19. Slušatelj eventa za pritisnutu tipku

```
window.addEventListener("keyup", function(e) {
    if(e.keyCode>=37 && e.keyCode<=40) { keysDown[e.keyCode] = false; }
});
```

Slika 20. Slušatelj eventa za tipku koja nije pritisnuta

Slike 19 prikazuje slušatelj eventa za tipku koja je pritisnuta i provjerava da li se radi o kodovima za tipke 37 do 40. Isto tako Slika 20 prikazuje isto to, samo za tipku ako nije pritisnuta.

```
if(!player.processMovement(gameTime) && gameSpeeds[currentSpeed].mult!==0)
{
    if(keysDown[38] && player.canMoveUp())          { player.moveUp(gameTime); }
    else if(keysDown[40] && player.canMoveDown())   { player.moveDown(gameTime); }
    else if(keysDown[37] && player.canMoveLeft())   { player.moveLeft(gameTime); }
    else if(keysDown[39] && player.canMoveRight())  { player.moveRight(gameTime); }
}
```

Slika 21. Provjera da li se igrač kreće.

Na Slici 21 se prikazuje da provjera da li je kretanje igrača i vrijeme u igri različito od nule, ako je istinito, onda se radi provjera da li su tipke „lijevo“, „gore“, „dole“ i „desno“ pritisnute i ako se igrač može kretati (ako se kreće prema zidu tada je provjera lažna), onda je provjera istinita te igrač se može pomaknuti za jedno mjesto na mapi.

Stvaramo objekt *viewport* koji označava vidno polje igre koje će pratiti sljedeće informacije: dužina i širina prikaza, koordinate gornje lijeve pločice koja je vidljiva (*startTile*) i koordinate daljnje desne pločice koja je vidljiva na mapi (*endTile*), istup pločica mape i objekata koji trebaju biti pomaknuti kada crtamo ekran relativno na njihovu prijašnju poziciju. Naše vidno polje ima *update* metodu koja uzima dva argumenta, x i y poziciju za koje želimo da nam budu u sredini vidnoga polja te radimo funkciju koja nam to računa. Brojevi se zaokružuju s *Math.floor* i daljnjim provjerama računamo i provjeravamo koordinate za *startTile* i *endTile*. Dio koda je prikazan na Slici 22.

```
//POCETNI PRIKAZ
var viewport = {
    screen      : [0,0],
    startTile   : [0,0],
    endTile     : [0,0],
    offset      : [0,0],
    update      : function(px, py) {
        this.offset[0] = Math.floor((this.screen[0]/2) - px);
        this.offset[1] = Math.floor((this.screen[1]/2) - py);

        var tile = [ Math.floor(px/tileW), Math.floor(py/tileH) ];

        this.startTile[0] = tile[0] - 1 - Math.ceil((this.screen[0]/2) / tileW);
        this.startTile[1] = tile[1] - 1 - Math.ceil((this.screen[1]/2) / tileH);

        if(this.startTile[0] < 0) { this.startTile[0] = 0; }
        if(this.startTile[1] < 0) { this.startTile[1] = 0; }

        this.endTile[0] = tile[0] + 1 + Math.ceil((this.screen[0]/2) / tileW);
        this.endTile[1] = tile[1] + 1 + Math.ceil((this.screen[1]/2) / tileH);

        if(this.endTile[0] >= mapW) { this.endTile[0] = mapW-1; }
        if(this.endTile[1] >= mapH) { this.endTile[1] = mapH-1; }
    }
};
```

Slika 22. Prikaz koda koji je zadužen za prikaz vidnog polja

5.2. Mehanike crtanja mape i ostalih objekata

Najlakši način za prikazivanje podataka i grafike na ekranu je da koristimo funkciju *requestAnimationFrame()* i da odredimo povratni poziv, odnosno da pokrenemo funkciju koja će se izvršiti kad je web preglednik spreman za prikazati podatke. Koristimo *window.onload = function()* a na Slici 22 je prikazan taj dio koda.

```
//FUNKCIJA KOJA UCITAVA PODATKE
window.onload = function()
{
    ctx = document.getElementById('game').getContext("2d");
    requestAnimationFrame(drawGame);
    ctx.font = "24pt bold monospace"; //STIL FONTA ZA PRIKAZ PODATAKA U IGRI
```

Slika 23. Isječak koda s funkcijom window.onload

Glavna funkcija koja će biti pozvana prijašnjom funkcijom, odnosno koja će raditi sljedeće naredbe je da crta mapu i da ponovno pozovemo funkciju kada bude web preglednik spreman za prikaz slike i podataka. Funkciju otvaramo tako da provjerimo možemo li nacrtati prikaz i ako ne možemo izlazimo iz funkcije i program završava. Prikazano je na Slici 24.

```
//CRTANJE ELEMENATA IGRE
function drawGame()
{

    if(ctx==null) { return; }
```

Slika 24. Prikaz početne funkcije s provjerom

Stvaramo par globalnih varijabli koje ćemo koristiti za crtanje mape i objekata. *Tile* označava jednu pločicu na ekranu i varijable *mapH* i *mapW* označavaju širinu i visinu mape. U *gameMap* stavljamo podatke koje su potrebni da bi se ta mapa prikazala na ekranu. Prikaz mape u kodu je na Slici 25.

```
//MAPA
var gameMap = [[
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,1,1,1,0,1,1,1,0,1,1,6,0,1,0,1,1,1,1,0],
    [0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,1,0,1,1,0],
    [0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,16,0,1,0],
    [0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1,1,0],
    [0,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,9,1,0],
    [0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,1,0,0,0,0],
    [0,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,1,0],
    [0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,1,0],
    [0,1,1,1,1,1,1,0,1,0,1,1,1,0,1,0,1,0,1,0],
    [0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,1,0],
    [0,5,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,1,0],
    [0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,1,1,0],
    [0,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,0],
    [0,1,0,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,0,0],
    [0,1,1,1,0,1,1,1,0,4,0,1,1,14,0,1,0,3,0,0],
    [0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,0],
    [0,1,8,1,1,15,0,1,1,1,7,1,0,1,1,1,10,1,1,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
],
```

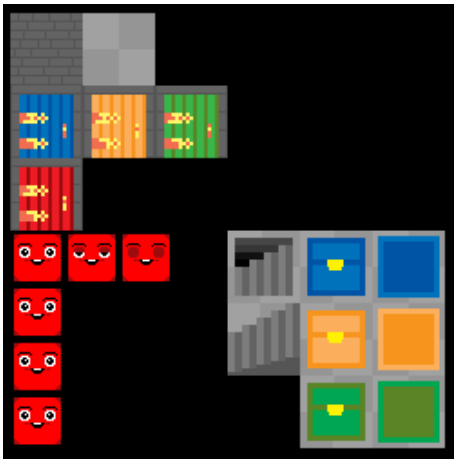
Slika 25. Prikaz mape prve razine

Na Slici 26 prikazane su varijable za širinu (*mapW*) i visinu (*mapH*) mape, kao i širina (*tileW*) i visina (*tileH*) pojedine pločice. Širina i visina se mjeri u pikselima.

```
var tileW = 40, tileH = 40;
var mapW = 19, mapH = 19;
```

Slika 26. Prikaz varijabli za širinu i visinu pločice i mape

Dodaju se varijable *tileset*, *tilesetURL* i *tilesetLoaded*. U *tilesetURL* varijabli doda se naziv slike koja služi toj varijabli da zna koje elemente treba uzeti iz slike. Slika 27 prikazuje datoteku s elementima koji se crtaju na zaslon.



Slika 27. Prikaz elemenata iz datoteke tileset

U varijablu *floorTypes* stavljaju se podaci o pločicama, označeni su brojevima od nula do trinaest. Varijabla *tileTypes* je varijabla u koju spremamo veličinu pločica koju nazivamo *sprite*. Svaki *sprite* je četrdeset piksela širok i četrdeset piksela dug. *Sprite-ove* „vadimo“ iz datoteke *tileset* i taj *sprite* se crta na ekranu. Na Slici 28 je prikazan *sprite* 0 koji predstavlja prvu pločicu u datoteci *tileset*. Ta pločica je zid.

```
var tileTypes = {
    0    : {floor:floorTypes.solid, sprite:[{x:0,y:0,w:40,h:40}]}, //ZID
```

Slika 28. Primjer pločice (sprite-a)

Lik *Qubo-a* je isto *sprite* i on je dimenzije trideset piksela dug i trideset piksela širok. Animacija *Quba* se postiže tako što se izmjenjuju sličice naizmjenično. Podatke koji se koriste su opisani u Slici 29. Podaci se nalaze u funkciji *Character*.

```
//QUBO
function Character()
{
    //QUBOVA POZICIJA
    this.tileFrom   = [1,1];
    this.tileTo     = [1,1];
    this.timeMoved  = 0;
    this.dimensions = [30,30];
    this.position   = [45,45];
    this.delayMove  = 150; //QUBOVA BRZINA

    this.direction  = directions.up;
    this.sprites = {};
    this.sprites[directions.up]     = [{x:0,y:120,w:30,h:30}, {x:30,y:120,w:30,h:30},
    {x:60,y:120,w:30,h:30}, {x:30,y:120,w:30,h:30}];
    this.sprites[directions.right]  = [{x:0,y:150,w:30,h:30}, {x:30,y:120,w:30,h:30},
    {x:60,y:120,w:30,h:30}, {x:30,y:120,w:30,h:30}];
    this.sprites[directions.down]   = [{x:0,y:180,w:30,h:30},
    {x:30,y:120,w:30,h:30}, {x:60,y:120,w:30,h:30}, {x:30,y:120,w:30,h:30}];
    this.sprites[directions.left]   = [{x:0,y:210,w:30,h:30},
    {x:30,y:120,w:30,h:30}, {x:60,y:120,w:30,h:30}, {x:30,y:120,w:30,h:30}];
}
```

Slika 29. Podaci za Qubo-vu animaciju

Na Slici 30 je prikazan dio koda koji omogućuje animaciju *Qubo-a*, u tom slučaju da trepne očima svakih par sekundi. Uzima podatke iz funkcije *Character* i izvodi se animacija nakon dvije sekunde a animacija traje jednu sekundu i dvjesto stotinki.

```
//QUBOVA ANIMACIJA
var sprite = player.sprites[player.direction];
ctx.drawImage(tileset,
    sprite[currentCharacterImage].x, sprite[currentCharacterImage].y,
    sprite[currentCharacterImage].w, sprite[currentCharacterImage].h,
    viewport.offset[0] + player.position[0], viewport.offset[1] + player.position[1],
    player.dimensions[0], player.dimensions[1]);

let canSwapImage = false;
const deltaT = timeStamp - lastSwapImageTimeStamp;

if (currentCharacterImage == 0 && deltaT >= 2000) {
    canSwapImage = true
} else if (currentCharacterImage > 0 && deltaT >= 120) {
    canSwapImage = true;
}

if(canSwapImage){
    currentCharacterImage = (currentCharacterImage+1)%4;
    lastSwapImageTimeStamp = Date.now()
}
```

Slika 30. Prikaz koda kojim se izvodi Qubo-va animacija

Podaci koji su prikazani na ekranu su razina i bodovi, varijabla koja to omogućuje zove se *ctx*. Na Slici 31 prikazan je dio koda koji ispisuje te podatke na gornjem lijevom kutu i boju fonta.

```
//BOJA FONTA
ctx.fillStyle = "black";
//PRIKAZ PODATAKA NA EKRANU (LEVEL)
ctx.fillText("RAZINA: " + (level+1), 5, 30);
ctx.fillText("BODOVI: " + score, 5, 60);
```

Slika 31. Prikaz koda za ispisivanje razine i bodova

U sljedećem isječku koda provjeravamo da li je učitavanje datoteke sa slikama bilo uspješno i provjeravamo sljedećom funkcijom koja je prikazana na Slici 32.

```
tileset = new Image();
tileset.onerror = function()
{
    ctx = null;
    alert("Failed loading tileset."); //PORUKA AKO SE NIJE UČITAO TILESET
};
```

Slika 32. Provjera da li je učitana datoteka

5.3. Mehanike dijaloških okvira i ostalih funkcija i varijabli

Implementacija modala ili dijaloškog okvira služi tome da, kada igrač stane na škrinju i otvori se dijaloški okvir s pitanjem i ponuđena su tri odgovora. Kada igrač stane na škrinju, izvodi se sljedeći kod koji je prikazan na Slici 33.

```
else if(tileFloor===floorTypes.question1)
{
    modal.getModal(1, 'predmet');
    currentSpeed = 1;
    count = count + 1;
    console.log(count);
    for(let i = 0; i < gameMap[level].length; ++i) {
        for(let j = 0; j < gameMap[level][0].length; ++j) {
            if(gameMap[level][i][j] === floorTypes.question1) {
                gameMap[level][i][j] = floorTypes.path1;
            }
        }
    }
}
```

Slika 33. Isječak koda u kojem se provjerava da li je pločica tipa question

Prvo što se provjerava je tip pločice i je li igrač stao na pločicu *question*, u ovom slučaju to je škrinja s pitanjem. Ako je provjera istinita, otvara se modal koji učitava podatke iz predmeta. Ovisno koji smo razred odabrali i koji smo predmet odabrali, program „vuče“ slučajnim odabirom pitanje iz *question.js*. Petljom se provjerava koordinata x i y u mapi i mijenja se tip pločice *question1* u tip pločice *path1*. Škrinja nestane i mijenja se u običan pod. Tako svaki put kada se točno odgovori na pitanje.

Da se pitanja prikažu u dijaloškom okviru moramo provjeriti lokaciju gdje se nalaze ta pitanja i uzmemo u obzir igračev odabir, odnosno koji je razred i predmet odabrao te mu postavljamo pitanje koje odgovara njegovom odabiru. Kako bi se elementi slike (*imgSrc*) i pitanje s ponuđena tri odgovora (*choiceA*, *choiceB* i *choiceC*) prikazala u dijaloškom okviru, trebamo ih također pretražiti u datoteci *questions.js*. Na Slici 34 se prikazuje gore navedeno.

```
class Modal {

    constructor() {

        this.modal = document.getElementById('modal');
    }

    getModal() {

        let params = new URLSearchParams(location.search);
        const razred = params.get('razred') ?? null;
        const predmet = params.get('predmet') ?? null;

        const { imgSrc } = questions['predmet'][predmet];
        const { question, choiceA, choiceB, choiceC, correct } = questions['predmet'][predmet][`${razred}r`].random();
```

Slika 34. Dio koda koji prikazuje pretraživanje elemenata za odgovore, sliku i pitanja

Zatim imamo funkciju *checkAnswer(correct)* koja provjerava da li je odgovor točan ili netočan. Najprije postavljamo brojač *score* na nulu. Ako je odgovor točan igrač dobiva sto bodova i aktivira se prikladan zvuk kojim igrač dobiva na znanje da je odgovor točan. Dijaloški okvir se zatvara. Mogućnost kretanja se nastavlja i igrač može nastaviti s igrom. Ako je odgovor netočan, igraču se oduzima pedeset bodova te se opet uključi prikladan zvuk koji igraču daje na znanje da je odgovor kriv. Dijaloški okvir se ne zatvara i postavi se novo pitanje (ne mora biti nasumično). Slika 35 prikazuje kod.

```
let score = 0;

function checkAnswer(correct){
    if( correct ){
        var audio = new Audio('correct.mp3');
        audio.play();
        score = score + 100;
        answerIsCorrect();
        currentSpeed = 0;
        this.modal.style.display = 'none';
    }else{
        var audio = new Audio('wrong.mp3');
        audio.play();
        score = score - 50;
        modal.getModal(1, 'predmet');
        currentSpeed = 0;
        this.modal.style.display = 'flex';
    }
}
```

Slika 35. Dio koda u kojem se provjerava je li odgovor točan ili netočan

Slika 36 prikazuje varijablu *gameSpeeds* u kojoj su pohranjene brzine koje omogućavaju da igrač stane prilikom obavljanja neke radnje ili opet dobije dozvolu za kretanje.

```
var gameSpeeds = [
    {name:"Normal", mult:1},
    {name:"Paused", mult:0}
];
```

Slika 36. Brzina kretanja igrača

Slika 37 prikazuje funkciju da ako igrač stane na pločicu *win*. Ako je funkcija istinita, tada se razina povećava za jedan. Slijedi daljnje provjeravanje da li je zadnja razina, ako je funkcija istinita, tada igra završava i ispisuje se poruka koja govori igraču da je završio igru i ispisuju mu se konačni bodovi. Ako nije zadnja razina tada se igrača smjesti na sljedeću razinu i na zadanim koordinatama. *Count* se postavlja na nulu, taj dio je vezan za drugu funkciju koja je naknadno objašnjena.

```
if(tileFloor===floorTypes.win)
{
    //BROJAC KOJI IDE ZA 1 LEVEL VISE
    ++level;

    //AKO JE ZADNJI LEVEL IGRA JE GOTOVA I VRACA SE NA INDEX.HTML
    if(level >= maxLevel) {
        alert("Bravo! \nDosao si do kraja igre!\nUkupan broj bodova: " + score);
        var audio = new Audio('end.mp3');
        audio.play();
        finalLevel();
    }
    else //AKO NIJE ZADNJI LEVEL QUBA POSTAVLJAMO NA ZADANE KOORDINATE
    {
        this.tileFrom   = [1,1];
        this.tileTo     = [1,1];
        this.position   = [45,45];
        count = 0;
    }
}
```

Slika 37. Dio koda koji je zadužen za provjeravanje ako igrač stane na pločicu "win"

Slika 38 prikazuje funkciju kojom se provjerava da li je *counter* došao do tri i pomoću *for* petlje provjeravaju se x i y koordinate pločice *door4* (crvena vrata) i zamjenjuje s pločicom *path1* (običan pod).

```
else if(count === 3)
{
    for(let i = 0; i < gameMap[level].length; ++i) {
        for(let j = 0; j < gameMap[level][0].length; ++j) {
            if(gameMap[level][i][j] === floorTypes.door4)
            {
                gameMap[level][i][j] = floorTypes.path1;
            }
        }
    }
}
```

Slika 38. Funkcija koja provjerava counter

Slika 39 prikazuje funkciju koja provjerava da li je igrač stao na gumb1 ili gumb2 ili gumb3. Ako je stao na jedan od tih gumba tada se izvrši funkcija *openDoor()*.

```
else if(tileFloor === floorTypes.gumb1 || tileFloor === floorTypes.gumb2 || tileFloor === floorTypes.gumb3) {
    openDoor(tileFloor, level);
}
```

Slika 39. Provjera da li je igrač stao na gumb1, gumb2 ili gumb3

Slika 40 prikazuje funkciju koja provjerava da li je objekt solidan ili nije, te ovisno o tome, može li igrač može navigirati kroz taj objekt.

```
Character.prototype.canMoveTo = function(x, y)
{
    if(x < 0 || x >= mapW || y < 0 || y >= mapH) { return false; }
    return !(tileTypes[gameMap[level][y][x]].floor !== floorTypes.path1 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.path2 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.gumb1 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.gumb2 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.gumb3 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.question1 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.question2 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.question3 &&
        tileTypes[gameMap[level][y][x]].floor !== floorTypes.win);

};
```

Slika 40. Provjera objekata da li se može prelaziti kroz njih

```
Character.prototype.canMoveUp       = function() { return this.canMoveTo(this.tileFrom[0], this.tileFrom[1]-1); };
Character.prototype.canMoveDown     = function() { return this.canMoveTo(this.tileFrom[0], this.tileFrom[1]+1); };
Character.prototype.canMoveLeft     = function() { return this.canMoveTo(this.tileFrom[0]-1, this.tileFrom[1]); };
Character.prototype.canMoveRight    = function() { return this.canMoveTo(this.tileFrom[0]+1, this.tileFrom[1]); };
Character.prototype.canMoveDirection = function(d) {
    switch(d)
    {
        case directions.up:
            return this.canMoveUp();
        case directions.down:
            return this.canMoveDown();
        case directions.left:
            return this.canMoveLeft();
        default:
            return this.canMoveRight();
    }
};

Character.prototype.moveLeft    = function(t) { this.tileTo[0]-=1; this.timeMoved = t; this.direction = directions.left; };
Character.prototype.moveRight   = function(t) { this.tileTo[0]+=1; this.timeMoved = t; this.direction = directions.right; };
Character.prototype.moveUp      = function(t) { this.tileTo[1]-=1; this.timeMoved = t; this.direction = directions.up; };
Character.prototype.moveDown    = function(t) { this.tileTo[1]+=1; this.timeMoved = t; this.direction = directions.down; };
Character.prototype.moveDirection = function(d, t) {
    switch(d)
    {
        case directions.up:
            return this.moveUp(t);
        case directions.down:
            return this.moveDown(t);
        case directions.left:
            return this.moveLeft(t);
        default:
            return this.moveRight(t);
    }
};
```

Slika 41. Dio koda koji pregledava igračevo kretanje

Slika 41 prikazuje da li se igrač, odnosno lik, može kretati po labirintu. Provjerava se za sve strane: lijevo, desno, gore i dolje. Ako mu je omogućeno kretanje, pozivaju se funkcije koje mu omogućuju kretanje na sljedeću pločicu u labirintu.

Slika 42 prikazuje slušatelja eventa kada je tipka pritisnuta. Kod tipke je 32 i ona označava tipku „*Space*“ na tipkovnici. Ako je zaista pritisnuta, tada se pojavi prozor koji daje upit korisniku da li želi izaći iz igre na početni zaslon.

```
if(e.keyCode==32) //TIPKA SPACE
{
    var r = confirm("Želiš li izaći na početnu stranicu?\nOK = Da\nCancel = Ne")
    if(r == true){
        resetGame();
    }
}
```

Slika 42. Provjera da li je pritisnuta tipka Space

Slika 43 prikazuje funkcijom kojom se provjerava da li je igrač stao na gumb. Ako je stao na gumb1 onda se otvaraju vrata1 (*door1*), ako je stao na gumb2 onda se otvaraju vrata2 (*door2*) i ako je stao na gumb3 onda se otvaraju vrata3 (*door3*). Za svaku provjeru gumba aktivira se zvuk. Na kraju funkcije je *for* petlja koja provjerava x i y koordinacije gdje se nalaze *door1*, *door2* i *door3* i zamjenjuje sa s *path1*.

```
function openDoor(buttonId, level) {

    let openDoor;

    switch(buttonId) {
        case floorTypes.gumb1:
            openDoor = floorTypes.door1;
            var audio = new Audio('door.mp3');
            audio.play();
            break;
        case floorTypes.gumb2:
            openDoor = floorTypes.door2;
            var audio = new Audio('door.mp3');
            audio.play();
            break;
        case floorTypes.gumb3:
            openDoor = floorTypes.door3;
            var audio = new Audio('door.mp3');
            audio.play();
            break;
        default:
            console.log('error');
            break;
    }

    for(let i = 0; i < gameMap[level].length; ++i) {
        for(let j = 0; j < gameMap[level][0].length; ++j) {
            if(gameMap[level][i][j] === openDoor)
            {
                gameMap[level][i][j] = floorTypes.path1;
            }
        }
    }
}
```

Slika 43. Prikaz funkcije koja otvara vrata

Zaključak

Na kraju ovog završnog rada, nakon analize dostupne literature na zadanu temu kao i izrada same didaktičke računalne igre, zaključuje se da je glavni cilj igara njihovo korištenje u edukacijske svrhe kod učenika od prvog do četvrtog razreda s ciljem poticanja razvoja njihova što kvalitetnijeg razmišljanja. Učenje im na ovaj način postaje zabavno, te uz igru im osigurava da uče efikasno. Djeca školske dobi kroz ovakvu vrstu igara razvijaju ponajprije svoje mentalne sposobnosti, a potom i psihomotorne. Igra je najbolji oblik učenja kroz zabavu koja djeci omogućuje brdo kreativnosti, sposobnost razmišljanja, logičkog zaključivanja, te na samom kraju, i rješavanje određenih prepreka, a uza sve to imajući za krajnji cilj: dolazak do kraja igrice.

Izradom igre u *JavaScriptu* naučio sam kako se koristi programski jezik *JavaScript* i koje su njegove prednosti i nedostaci te za koje vrste aplikacija je najviše pogodan za razvijanje. Osim toga koristio sam i *HTML5* za izradu stranice te *CSS* za oblikovanje stila stranice. Naučio sam kako napraviti funkcionalnu web aplikaciju kombiniranjem ova tri elementa.

Ova tema mi je bila dobra prilika, odnosno izazov da naučim sve što treba znati o izradi didaktičke računalne igre i kako najefikasnije implementirati elemente edukacije u računalnu igru i da pritom bude učeniku zanimljiva prilikom korištenja. Naučio sam kroz čitanje literature kako su se razvijale didaktičke računalne igre kroz povijest, koji se pristupi koriste tijekom njihovog razvoja i što sve ulazi u sam proces njihove izrade.

Popis literature

[1] Digitalne igre u nastavi // Dostupno na: http://webfestival.carnet.hr/2014_digitalne_igre_u_nastavi (09.09.2020.)

[2] Stevanović, M. (2003). Predškolska pedagogija. Rijeka; Andromeda, str. 121.

[3] *Games as teaching method*. *Cometa Ressearch* // Dostupno na: http://cometaresearch.org/tag/didactic-games/#:~:text=A%20didactic%20game%20is%20construed,achieve%20a%20strictly%20defined%20score.&text=Examples%20of%20didactic%20games%20strengthening,based%20games%20and%20strategic%20games (09.09.2020.)

[4] K. Corti, "*Games-based Learning; a serious business application", Inf. PixelLearning*, vol. 34(6), pp. 1–20, 2006. (09.09.2020.)

[5] Didaktika // Dostupno na: https://hr.wikipedia.org/wiki/Didaktika (24.08.2020.)

[6] Didaktika, Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2020 // Dostupno na: http://www.enciklopedija.hr/Natuknica.aspx?ID=14982 (24.08.2020.)

[7] *Game-Based Learning Vs. Gamification: Do You Know The Difference*? *Learning Technologies*. // Dostupno na: https://trainingindustry.com/articles/learning-technologies/game-based-learning-vs-gamification-do-you-know-the-difference/ (09.09.2020.)

[8] *Is Gamification the Only Way for Apps to Survive*? *GetSocial*. // Dostupno na: https://blog.getsocial.im/is-gamification-the-only-way-for-apps-to-survive/ (09.09.2020.)

[9] Franković, I. Učenje temeljeno na didaktičkim računalnim igrama. // Dostupno na: https://www.inf.uniri.hr/images/studiji/poslijediplomski/kvalifikacijski/Frankoivc_Ivona_Kvalifikacijski_rad.pdf (09.09.2020.)

[10] *Educational video game*. Wikipedia. // Dostupno na: https://en.wikipedia.org/wiki/Educational_video_game (09.09.2020.)

[11] Digitalne igre u školama: Priručnik za učitelje. // Dostupno na: http://os-bukovac-zg.skole.hr/upload/os-bukovac-zg/images/static3/2252/attachment/1437640603_04-487-digitalne-igre-u-skoli-prirucnik-za-ucitelje.pdf (09.09.2020.)

[12] *Logo* (programming language) // Dostupno na: https://en.wikipedia.org/wiki/Logo_(programming_language)#cite_note-17 (25.08.2020.)

[13] *Turtle graphics* // Dostupno na: https://en.wikipedia.org/wiki/Turtle_graphics (25.08.2020.)
[14] *LEGO/Logo* // Dostupno na: https://link.springer.com/chapter/10.1007%2F978-3-662-02938-1_2 (28.08.2020.)
[15] *ObjectLOGO* // Dostupno na: https://codelani.com/languages/objectlogo.html (25.08.2020.)
[16] *Atari Logo* // Dostupno na: https://en.wikipedia.org/wiki/Atari_Logo (25.08.2020.)
[17] *The History of the Educational Gaming Industry*. *Toki Toki*. // Doustupno na: https://www.tiki-toki.com/timeline/entry/176491/The-History-of-the-Educational-Gaming-Industry#vars!panel=1681142! (28.08.2020.)
[18] *Math Blaster*! // Dostupno na: https://en.wikipedia.org/wiki/Math_Blaster! (26.08.2020.)
[19] *Where in the World is Carmen Sandiego* (1985 video game) // Dostupno na: https://en.wikipedia.org/wiki/Where_in_the_World_Is_Carmen_Sandiego%3F_(1985_video_game) (26.08.2020.)
[20] *Elmo`s Preschool* // Dostupno na: https://muppet.fandom.com/wiki/Elmo%27s_Preschool (26.08.2020.)
[21] *CD* Učilica. Učilica. // Dostupno na: https://www.ucilica.tv/ucilica (26.08.2020.)
[22] *National Geographic Challenge*. *Educate The Magazine for Schools*, *Parents and Pupils*. // Dostupno na: http://www.educatemagazine.com/national-geographic-challenge/ (26.08.2020.)
[23] *World Rescue*. *World Rescue Game*. // Dostupno na: http://worldrescuegame.com/ (27.08.2020.)
[24] *Prodigy Math Game*. *Google Play*. // Dostupno na: https://www.parentingscience.com/educational-video-games.htmlhttps://play.google.com/store/apps/details?id=com.prodigygame.prodigy&hl=en (27.08.2020.)
[25] *Educational video games*: *A guide for the science-minded*. *Parenting Science*. // Dostupno na: https://www.parentingscience.com/educational-video-games.html (09.09.2020.)
[26] *JavaScript Tutorial*. *W3schools.com*. // Dostupno na: https://www.w3schools.com/js/ (01.08.2020.)

[27] *JavaScript*. *Wikipedia*. // Dostupno na: https://en.wikipedia.org/wiki/JavaScript (02.08.2020.)

[28] *The History of JavaScript*. *Springboard Blog*. // Dostupno na: https://www.springboard.com/blog/history-of-javascript/ (04.08.2020.)

[29] *Java Vs*. *JavaScript*: *A Short Comparison*. // Dostupno na: https://scand.com/company/blog/java-vs-javascript/ (09.09.2020.)

[30] Uvod u *JavaScript*: *JavaScript Introduction*. MojWebDizaj. // Dostupno na: https://www.mojwebdizajn.net/web-programiranje/vodic/javascript/uvod-u-javascript.aspx (09.09.2020.)

[31] *Can JavaScript Be Used To Make Games*? *JSDiaries*. // Dostupno na: https://www.jsdiaries.com/can-javascript-be-used-to-make-games/ (09.09.2020.)

[32] *A List of Awesome Games Made With Html5 and Javascript*. *Ma-no*. // Dostupno na: https://www.ma-no.org/en/programming/javascript/15-awesome-games-made-with-html5-and-javascript (09.09.2020.)

Popis priloga

Popis slika:

Popis tablica:

Popis skica: