Blockchain i video igre

Posavac, Matej

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:623282

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

BLOCKCHAIN I VIDEO IGRE

Završni rad

Matej Posavac

Osijek, 2019.

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. IGRE I KRIPTOVALUTE 2
2.1. CryptoKitties 2
2.2. EOS.IO platforma 4
2.2.1. EOS Knights 5
2.2.2. Prospectors 6
3. DRUGAČIJA UPOTREBA BLOCKCHAINA 7
3.1. eSport i DreatmTeam.gg 7
3.2. Nove mogućnosti 8
3.2.1. HashCraft 8
3.2.2. CryptoZombies 9
3.3. Razvoj u budućnosti 10
4. BLOCKCHAIN ZMIJICA 11
4.1. Korištene tehnologije 11
4.2. Dizajn igre 12
4.3. Pametni ugovori 16
4.4. Implementacija pametnog ugovora 17
4.4.1. Solidity 17
4.4.2. Web3 19
4.4.3. Povezivanje sa igrom 21
5. ZAKLJUČAK 25
LITERATURA 26
SAŽETAK 28
ABSTRACT 29
ŽIVOTOPIS 30
PRILOZI 31

1. UVOD

Blockchain je u današnje vrijeme sve prisutnija i brzo rastuća tehnologija. Danas je blockchain najprisutniji u obliku kriptovaluta, ali sve više pronalazi primjenu u raznim područjima upravo zbog svoje sigurnosti, načina rada i kvalitete.

U drugu ruku industrija video igara iz godine u godinu doživljava rast te je prošle godine generirala preko 130 milijardi američkih dolara, pretekavši tako glazbenu i filmsku industriju. Igre se danas pojavljuju i igraju na sve većem broju raznih platformi poput mobitela, virtualne stvarnosti i ostalih konzola, a eSport (engl. *Eletronic Sport*) postaje sve unosnije zanimanje.

U ovom ću završnom radu pokazati kakvo je trenutno stanje industrije blockchain video igara, koja je tek u svojim začetcima. U prvom poglavlju će biti opisano koje su prednosti i nedostatci korištenja određenog oblika kriptovaluta u igrama.

Sljedeće poglavlje će opisati na koje se druge načine trenutno koristi blockchain u igrama, te na koje je još načine u budućnosti moguće iskoristiti potencijal blockchain tehnologije.

Za kraj ću kroz primjer JavaScript igre Zmijica pokazati kako se blockchain može iskoristiti za spremanje rezultata igre koristeći koncept pametnih ugovora (engl. *Smart Contracts*) i programskog jezika Solidity za implementaciju pametnih ugovora.

1.1. Zadatak završnog rada

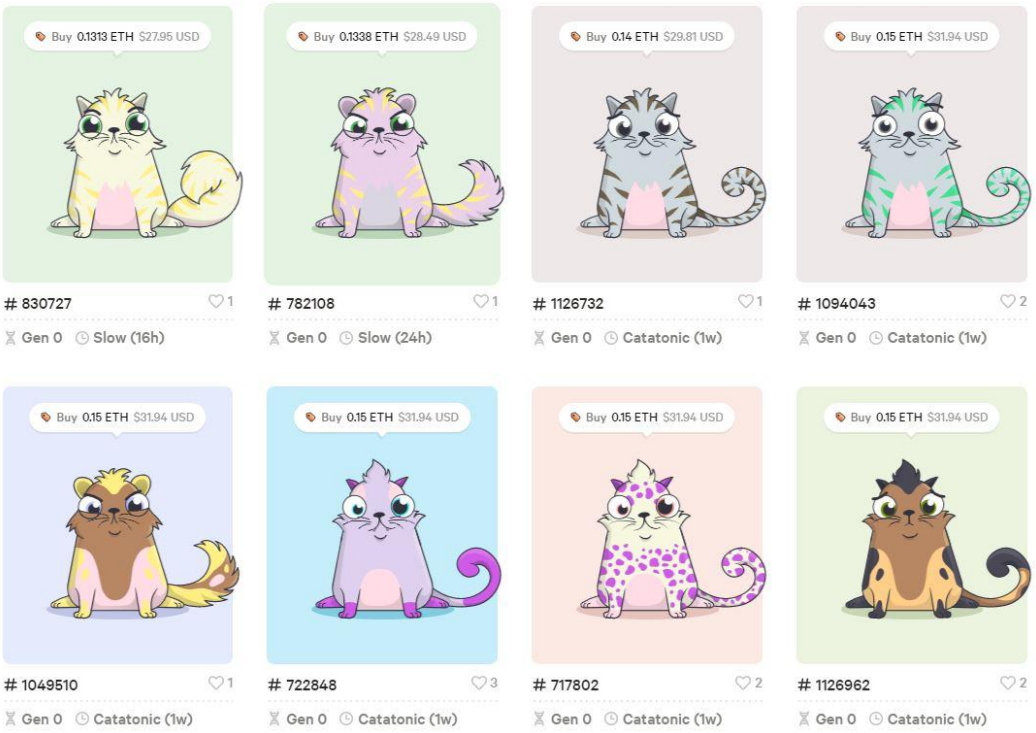
U radu je potrebno opisati primjenu blockchain tehnologije u industriji video igara i na koje sve načine je moguće ove dvije tehnologije povezati. Također je potrebno i dati neke nove ideje u ovom području.

2. IGRE I KRIPTOVALUTE

Nakon iznimnog uspjeha Bitcoin kriptovalute, mnogi su u blockchainu vidjeli mogućnost zarade. Ethereum blockchain platforma specijalizirala se za decentralizirane aplikacije i time otvorila mogućnost mnogim programerima za razvoj blockchain igara i tako postala jedna od najkorištenijih i vrjednijih blockchain mreža. Kroz ovo poglavlje biti će opisane najpopularnije i najuspješnije igre koje kao svoju bazu imaju korištenje kriptovalute.

2.1. CryptoKitties

CryptoKitties je igra razvijena za Ethereum blockchain platformu. Igra je osmišljena i postavljena na platformu u listopadu 2017. godine. Igrači imaju mogućnost sakupljanja, uzgajanja, kupovine i prodaje stvorenja zvanih *CryptoKitties* odnosno digitalnih mačaka. Svaka digitalna mačka ima jedinstvene osobine i izgled koji su definirane njenim genima. Također ima mogućnost razmnožavanja čime se dobiva novi jedinstveni potomak čiji se genski zapis čuva na blockchainu. Tvorci igre su od trenutka stvaranja igre do studenog 2018. godine objavili 50 000 mačaka i svaku od njih je mogao kupiti isključivo jedan igrač. Te mačke su kategorizirane kao mačke nulte generacije, a njihovim parenjem nastaju mačke prve generacije i tako se svakim potomkom generacije dalje razvijaju. Dobiveni potomak se dalje može uzgajati, prodati ili postaviti na aukciju.



Slika 2.1. Sučelje CryptoKitties igre

Igra je otvorenog koda i sadrži nešto više od 2000 linija koda. *CryptoKitties* je napisana u Solidity programskom jeziku i pohranjena unutar Ethereum blockchain platforme te na taj način svaki čvor sadrži kopiju koda. Sama igra je podijeljena u nekoliko pametnih ugovora koji su međusobno povezani nasljeđivanjem.

Promatrajući pametne ugovore izvornog koda [2] vidljivo je kako su definirana osnovna pravila za pristup određenim akcijama *CryptoKittiesa.* Ugovor za kontrolu pristupa sadrži adrese odgovornih koji mogu izvršavati akcije. Na primjer glavni izvršni direktor može dodjeljivati uloge i staviti na čekanje pametni ugovor odnosno pauzirati ga. Kada se takva akcija pokrene korisnici više ne mogu prodavati, kupovati niti dalje razmnožavati digitalne mačke. Adresa direktora se inicijalno postavlja na onu adresu koja je i stvorila ugovor. Nadalje, financijski direktor može pokretati akcije vezane za financijski i aukcijski aspekt. Šef odjela ima mogućnost objavljivanja aukcija mačaka nulte generacije i objave promotivnih mačaka.

Poseban ugovor je zadužen za način na koji su podaci o korisniku i mačkama pohranjeni. Ugovor prvo definira mačku i to kao strukturu sačinjenu od 8 varijabli tipa uint (*unsigned integer*) za svaki od važnih parametara poput gena mačke, koji nam govori kako će mačka izgledati, vrijeme rođenja, vrijeme nakon kojeg se mačka može razmnožavati, generaciju mačke i drugo. Također ugovor definira polje sa svim mačkama koje postoje unutar igre. Definira i tko je vlasnik pojedine mačke.

Ugovor koji vodi računa o vlasništvu nasljeđuje ERC-721. To je standard namijenjen za stvaranje digitalnih ne zamjenjivih tokena koji predstavljaju neku kolekciju. On osigurava jedinstvenost i cijenu koja je vezana za tu jedinstvenost. Implementacijom ERC-721 funkcija omogućuje se korisniku prebacivanje tokena nekom drugom igraču komunikacijom izravno sa pametnim ugovorom.

Kao što je već navedeno digitalne mačke se mogu i razmnožavati i to se može učiniti na dva načina. Vlasnik međusobno može upariti dvije mačke koje posjeduje ili ženku može upariti sa javno dostupnim mačkom. Potomak će nositi broj generacije roditelja uvećan za jedan. Aukcijom se također omogućuje stavljanje mačke na tržište i na taj način omogućuje drugim igračima da pare vlastitu mačku sa njom. Poseban ugovor zadužen je za implementaciju svih metoda potrebnih za uspješno razmnožavanje. No ugovor koji odgovoran za genetska obilježja potomka nije javno objavljen jer bi u suprotnom svima bile dostupne formule kojima se određuje vrijednost i rijetkost pojedine digitalne mačke i na taj bi način igrači znali koje mačke treba međusobno pariti kako bi se dobio najvrjedniji potomak. Nakon što je dobiven potomak, roditeljima se postavlja određeno

vrijeme za koje se ne mogu ponovno pariti. Na posljetku glavni ugovor igre nazvan je „KittyCore“ i nasljeđuje sve stvorene ugovore. Uz to omogućuje konačno dohvaćanje svih podataka vezanih za digitalnu mačku pomoću njene identifikacijske oznake.

No igra nije u potpunosti decentralizirana, odnosno ne nalaze se svi njeni podaci na Ethereum blockchain platformi. Unutar Solidity koda ne nalazi se niti jedna varijabla koja čuva zapis o slici i opisu pojedine mačke već samo genetski zapis digitalne mačke. Sve slike mačaka pohranjene su na *CryptoKitties* serveru i to iz razloga što bi spremanje svih tih informacija u blokove na blockchain dovelo do preopterećenja cijele platforme čime bi se usporio i zagušio promet odvijanja transakcija u cijelom sustavu.

Svakom kupnjom, razmnožavanjem, stvaranjem, odnosno za svaku akciju, potrebna je određena količina *gasa*. *Gas* je naknada koju svaki igrač mora plaćati kako bi se određeni zahtjev mogao poslati rudarima Ethereum mreže, odnosno na svaki čvor blockchaina. Svaka digitalna mačka ima svoju cijenu. „Genesis“ je prva kripto mačka i prodana je svome vlasniku za 250 Ethera odnosno otprilike 120 000 američkih dolara [3], no to i dalje nije najvrjednija mačka, naime najskuplja mačka je „Dragon“ i kupljena je za 600 Ethera odnosno oko 170 000 američkih dolara.[4] *CryptoKitties* je u trenutku predstavljanja bio revolucionaran koncept i nakon njenog uspjeha mnogi tvorci igara su se okrenuli upravo takvim igrama na blockchainu. Digitalne mačke su zamijenjene drugim predmetima, Ethereum platforma drugim mrežama, no srž igara je ostala ista.

2.2. EOS.IO platforma

Poput Ethereuma, EOS.IO je blockchain platforma namijenjena decentraliziranim aplikacijama i igrama. Kada pogledamo ljestvicu popularnosti blockchain igara prema broju korisnika, uočljivo je kako ih 4 koriste EOS.IO platformu, dok ih 5 koristi Ethereum platformu. No ukoliko pogledamo rangiranje prema broju transakcija jasno se može vidjeti da EOS.IO koristi 7 od 10 igara nasuprot nula igara na Ethereumu.[5] Korisnik pri svakoj akciji na Ethereumu mora platiti određenu količinu *gasa,* odnosno naknade. U drugu ruku, EOS.IO programerima daje mogućnost ulaganja tokena za korištenje aplikacije i kao rezultat korisnik ne plača naknadu za izvršavanje akcije.

Razlike između platformi su izrazito uočljivi jer se zapravo radi o različitim ideologijama. Ethereum platforma je mnogo sporija i može obraditi manje transakcija u sekundi jer koristi *Proof-of-Work* algoritam. Algoritam funkcionira na način da svi čvorovi, odnosno rudari unutar lanca (engl. *Miners*), rješavaju kriptografski problem i onaj koji prvi pronađe rješenje dobiva nagradu.

Ovim načinom oni rudari koji s vremenom rastu ili se udružuju u zajednice kako bi što prije riješili problem, postaju opasnost za decentralizaciju. EOS.IO se pak temelji na *Proof-of-Stake* algoritmu. Taj algoritam nasumično izabire čvor koji će potvrditi sljedeći blok (engl. *Validators).* Kako bi postao validator čvor mora uložiti određenu količinu novca u sam sustav koji će izgubiti ukoliko odobri prevarantsku transakciju. Korištenjem delegiranog (engl. *Delegated) Proof-of-Stake* algoritma, EOS.IO ima određen broj validatora za koje se glasa što ponovno može predstavljati opasnost za decentralizaciju. Korištenjem ovog algoritma postiže se mogućnost obrade više transakcija u sekundi.

Razlike su uočljive i u samom postavljanju pametnih ugovora. Jednom postavljeni ugovor na Ethereum je neizmjenjiv, dok EOS.IO omogućava nadogradnju i izmjenu ugovora što predstavlja jednu vrstu sigurnosnog rizika. Zbog svoje mogućnosti obrade više tisuća transakcija u sekundi, igre objavljenje na EOS.IO platformi trenutno drže sami vrh ljestvica po broju korisnika.

2.2.1.EOS Knights

Jedna od najpopularnijih igra na EOS.IO platformi je EOS Knights.[5] Igra se zasniva na tri lika, vitezu, strijelcu i čarobnjaku čija je uloga zaštiti svoje selo od napada. Oni brane selo sve dok im ne ponestane energije nakon čega ih je potrebno ponovno oživiti. U tim napadima im pomažu i ljubimci koje je moguće dobiti kroz dodatne transakcije. Ubijanjem neprijatelja napreduje na nove razine. Što je veća razina veće su i šanse osvajanja rijetkih i jedinstvenih predmeta i materijala. Osvajanjem materijala mogu se kreirati predmeti koji će likove učiniti jačima ili se isto tako kreirani predmeti mogu prodati drugom igraču. Ukoliko se odluči na prodaju, igrač određuje cijenu predmeta kojeg želi prodati i to u EOS kriptovaluti. Nakon što se drugi igrač odluči za kupnju tog predmeta transakcija se izvršava na blockchainu i on postaje novi vlasnik predmeta kojeg mu nitko ne može otuđiti. Igrači na taj način mogu i zaraditi, ali naravno i potrošiti ukoliko žele brži napredak u igri.

Ono zbog čega je igra izrazito popularna je mogućnost igranja i preko *Android* i *IOS* pametnih mobitela, kao i rangiranje igrača čime se stvara dodatna kompetitivnost među igračima. Može se zaključiti kako se blockchain u EOS Knightsu koristi isključivo za provođenje prodaje i kupnje predmeta u igri.

2.2.2. Prospectors

Trenutno najpopularnija igra na ljestvicama je *Prospectors* i to sa više od 2800 dnevnih korisnika na dan 30.7.2019.[5] Igrač igru započinje na lokaciji sa tri radnika koji igraču omogućuju izgradnju raznih objekata. Za razliku od ostalih igara *Prospectors* se istinski može igrati. Igrač može izgraditi razne kuće, staje i ostale objekte koje otključava svojim razvojem, također može posjećivati ostale lokacije koje mogu biti bogate sa raznim resursima potrebnim za izgradnju određenih objekata. Radnici mogu istraživati kartu, pronalaziti materijale i resurse, transportirati materijale drugim igračima i sve to kako bi zaradili zlato koje je zapravo kriptovaluta unutar igre. To zlato omogućuje kupnju drugih alata za radnike, materijale, lokacije, drugim riječima koristi se kao baza za odnose među igračima. Igrači se mogu udruživati i u organizacije kako bi bili efikasniji u svome napretku. *Prospectors* je igra koja podigla ljestvicu što se tiče kompleksnosti uporabe blockchaina. Svaki predmet koji se može prodati i tako zamijeniti za zlato kao i akcija koja se napravi sa tim predmetom se bilježi na blockchain što ovu igru stavlja na prvo mjesto, ne samo glede broja korisnika već i po broju transakcija kojih je unutar tjedan dana zabilježeno preko 425 000 [5], što je rekordno u usporedbi za ostalim blockchain igrama.

3. DRUGAČIJA UPOTREBA BLOCKCHAINA

Iako je blockchain pronašao svoju primjenu, ponajviše u području kriptovaluta i novca, postoje primjeri gdje bi se blockchain mogao iskoristit za unaprjeđenje eSporta i istinskog decentraliziranog igranja igara.

3.1. eSport i DreamTeam.gg

eSport je natjecanje u video igrama između profesionalnih igrača. Takva vrsta natjecanja se počela razvijati davnih 80-tih godina kada su se igrači natjecali na turnirima u arkadnim igricama. Danas je to postalo globalno natjecanje gdje se igrači natječu u *multiplayer* igrama poput *League of Legends, Counter-Strike* ili *Fortnite*. Koliko je eSport uzeo zamaha može se vidjeti i iz činjenice da je međunarodni olimpijski odbor razmatrao odluku da se uvede kao sastavni dio ljetnih igara 2024. godine u Parizu. U konačnici je zaključeno kako je ipak pre rano za uvođenje. Sa daljnjim razvojem eSporta sve su izraženiji i problemi zajednice igrača. Od samog procesa sastavljanja idealnih timova pa sve do ne isplaćivanje nagrada i plača igračima. Mali postotak ukupne zarade generirane eSportom ide izravno igračima. Kako bi se eSport nastavio razvijati turniri se trebaju približiti igračima, ali problem je povjerenje koje turniri gube neisplatom nagrada. Čak i pobjeda na turnirima ne garantira igračima da će im biti isplaćena nagrada. Igrači mogu čekati i po četiri mjeseca na isplatu, što je posebno problematično ukoliko igrači moraju platiti naknadu za pristup samom turniru. Nadalje, eSport industrija je poprilično ne povezana, odnosno ne postoji jedinstveno mjesto za okupljanje timova, pronalaženje sponzora, prikupljanje statističkih podataka o napretku igrača, itd.

DreamTeam je uvidio da postoje problemi za čije rješenje bi se mogli iskoristi blockchain i pametni ugovori. DreamTeam je platforma za pronalaženje timova, sponzora, upravljanje transakcija igrača i skupljanje i analizu statističkih podataka o napretku igrača. Trenutno podržavaju igrače igara *League of Legends, Counter-Strike* i *Apex* [6], međutim planiraju proširenje i podršku za ostale popularne igre. Platforma pruža mogućnosti i objedinjuje alate na kojima sponzori, igrači, timovi i turniri mogu zajedno surađivati. DreamTeam posjeduje i vlastitu valutu u kojoj se isplaćuju bonusi za igrače, sponzorski ugovori, naknade za pristup turniru, i ono najvažnije donacije i novčane nagrade turnira. Osim isplate u vlastitoj valuti, omogućuje se zaključivanje ugovora gdje se isplaćuje fiksni iznos u američkim dolarima. Nakon što igrač pobjedi na turniru, unutar nekoliko minuta se automatizmom može izvršiti isplata nagrade. Sve to zasniva se na pametnim ugovorima i to na način da se novčane nagrade turnira prebace i zadrže na pametnom ugovoru sve dok se ne dobije pobjednik turnira. Nakon pobjede tima, novčana se

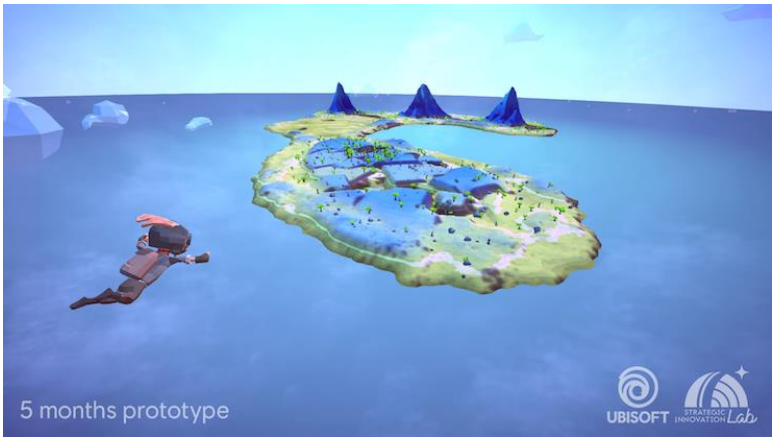
nagrada automatski prenosi pobjedničkom timu te se nagrada pravedno i jednako raspodjeljuje među igračima tima. U srpnju 2019. godine DreamTeam je bilježio više od 700 000 registriranih igrača sa daljnjim ambicijama za rast.[6]

3.2. Nove mogućnosti

3.2.1. HashCraft

Koliko potencijala se krije iza blockchaina igara govori i činjenica kako se div industrije igara Ubisoft uključio u njihov razvoj. HashCraft je još u ranoj fazi razvoja no nudi jednu drugačiju perspektivu i pogled na blockchain. Kako ime i prototip sugeriraju, na igru su izravno utjecali trenutno mega popularni naslovi poput *Minecrafta* i *Fortnitea.*

Na početke igre, automatski se generira jedan otok na koji igrač slijeće i upravo ovdje se uključuje blockchain što sugerira i *hash* iz naziva igre. Svi podaci otoka se ne spremaju tradicionalno na server ili lokalno, već se pohranjuju na blockchain. Igrač se slobodno može kretati po otoku, istraživati ga i u konačnici izgraditi onakav svijet kakav je zamislio. Još uvijek nije u potpunosti jasno kako će cijeli koncept funkcionirati, no predstavljen je na način da će se svaka promjena na otoku pohraniti na blockchain i jednom kada je igrač stvorio otok kao i zagonetke i izazove na njemu, otvoriti će se mogućnost javnog objavljivanja otoka. Iako to nije glavna vodilja igre, osmislili su i kriptovalutu kojom će se moći, ukoliko igrač to želi, naplaćivati naknada za ulaz na otok drugim igračima kao i nagrađivati one koji uspješno izvrše izazove na otoku. Nagrada ne mora biti isključivo novčana u obliku kriptovalute već može biti i iznimno rijedak predmet iskoristiv unutar igre. S obzirom da se podaci o igri ne nalaze na serveru već na svakom čvoru, njih se ne može hakirati ili obrisati. Pohranom svih podataka na blockchain postiže se da istinsko vlasništvo nad svijetom i vlastitim podacima pripada izravno igračima, a ne tvorcu igre kako je to do sada bilo.

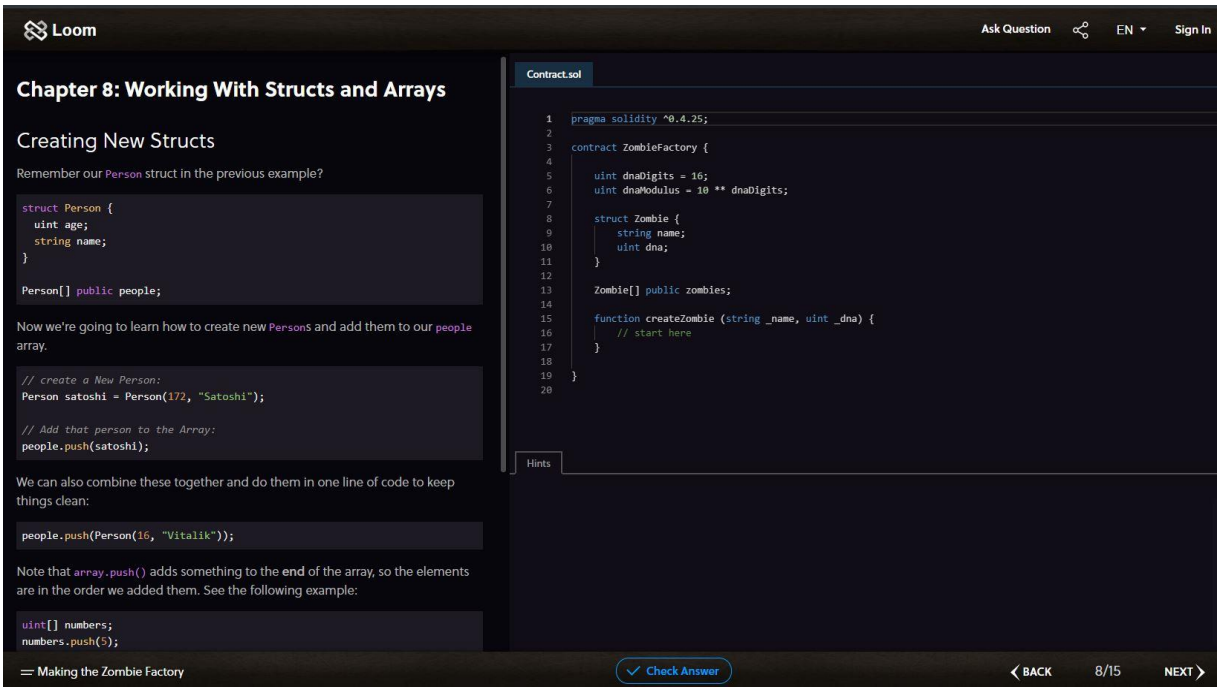


Slika 3.1. Prototip HashCrafta, Ubisoftove blockchain igre, izvor: [8]

3.2.2. CryptoZombies

Iako bi se iz naslova moglo zaključiti kako je riječ o još jednoj igri koja slijedi uspjeh *CryptoKittiesa*, zapravo je riječ o hvale vrijednim i besplatnim lekcijama za izradu blockchain igre za Ethereum platformu. Korisniku se kroz niz lekcija objašnjavaju osnove rada sa Solidity programskim jezikom, pametnim ugovorima, standardima poput ERC-721 kao i Web3 JavaScript biblioteke za interakciju sa Ethereum platformom. Lekcije nisu samo pusto pisanje koda, već se kroz interakciju i prikaz onoga što je korisnik napravio uistinu može vidjeti rezultat kreiranja igre. Prva lekcija kreće od opisivanja proces nasumičnog generiranja zombija i kroz lekcije objavljivane na tjednoj bazi nadograđuje igru dodavanjem borbe sa drugim zombijima i ostalim mogućnostima.

Nakon što uspješno završi svaku od lekcija korisnik za nagradu dobiva jednog zombija kojeg, nakon što završi igru i objavi ju na mreži, može iskoristiti za borbu protiv drugih igrača u stvarnoj blockchain igri. Loom Network je tvorac ove ideje, a poticaj koji se krije iza nje je edukacija i inspiracija programera za razvoj blockchain igara jer smatraju kako je to način na koji se može unaprijediti industrija video igara. [9]



Slika 3.2. Sučelje CryptoZombies sustava za izradu igre

3.3. Razvoj u budućnosti

Kao što je vidljivo iz primjera svih trenutno postojećih blockchain igara one se zapravo svode na kriptovalute i tokene, što je mnogima još uvijek ne predvidljivo i ne stabilno područje. Uz poneku uspješnu priču o zaradi igrača, većinom je kao i u klađenju tvorac taj koji najviše profitira od tih igara. Igrači te igre trenutno koriste za skupljanje raznih kolekcija, prodaju i razmjenu stvari, a ne uistinu za igranje sa njima.

Blockchain tehnologija itekako ima budućnosti u ovom području. Mogao bi biti rješenje za pitanje rangiranja igrača preko više različitih platforma i to bez mogućnosti manipuliranjem sa tim ljestvicama. Statistički podaci o profesionalnim igračima bi se sigurno mogli spremati na blockchain čime bi postali transparentniji, dostupniji svima i lišeni svake mogućnosti prevare nad njima.

Koncept i ideja tvrtke Ubisoft i igre *HashCraft* otvaraju nove mogućnosti primjene blockchaina kroz igru. Umjesto baziranja na kriptovalutama, sljedeći primjer možemo sagledati i kroz razvoj igre provjerom povijesti transakcija. Igrač na početku igre posjeduje oružje, to oružje napreduje kao što i igrač napreduje u igri. Umjesto da ga se iskoristi za prodaju i razmjenu pomoću kriptovaluta mogao bi se razvijati na način da prolazi od igrača do igrača i postaje sve snažniji sve dok u konačnici ne postane najsnažnije oružje od čega bi svi koji su ga ranije koristili mogli imati određenu korist. U ovom bi se primjeru upotrebom blockchaina i povijesti transakcija doista moglo utvrditi tko su mu bili prethodni vlasnici. Drugi primjer gdje bi ovakav koncept bio od osobite koristi je kroz natjecanje gdje je potrebno vidjeti slijed igranja igrača. U momentnim igrama gdje je bitno tko je prvi pucao, ukoliko se to uistinu zapisalo u bloku, moglo bi se doista i autorizirati tko je pucao prvi. Također virtualni objekti osvojeni i zarađeni u jednoj igri bi mogli pronaći svoju svrhu i u drugoj igri zahvaljujući mogućnostima blockchaina.

Zajednica programera blockchain igara iz dana u dan je sve veća i tomu dakako doprinosi i *Blockchain Game Alliance*. To je organizacija čiji je cilj promicanje razvoja blockchaina u industriji video igara. To čine na način da educiraju i šire svjesnost medu zajednicama i programerima o potencijalima koje blockchain pruža svojom implementacijom u videoigre. Organiziraju razne konferencije i događaje te povezuju članove preko internetskih servisa kroz koje mogu učiti ili pomoći drugima u razvoju. Članstvo u organizaciji je trenutno besplatno i jedan je od pozitivnih primjera kako zajednice rade na unaprjeđenju blockchaina u videoigrama.

4. BLOCKCHAIN ZMIJICA

U ovom će poglavlju na primjeru JavaScript igre zmijica biti pokazano kako se blockchain koncept može iskoristiti na primjeru konkretne igre gdje se korištenjem pametnih ugovora i programskog jezika Solidity najbolji rezultat i ime igrača spremaju na Ethereum blockchain.

4.1. Korištene tehnologije

Visual Studio Code je Microsoftov uređivač izvornog koda. Iznimno je lagan za instalaciju i jednostavan za korištenje. Omogućuje integriranu podršku za rad sa JavaScriptom, TypeScriptom i Node.jsom. Kroz mnoge dodatke i nadogradnje omogućuje dodatnu podršku za veliki broj jezika kao što su C++, C#, Java, Python, PHP i još mnogi drugi.[11]

Remix je alat otvorenog koda napisan u JavaScriptu koji omogućuje pisanje Solidity programskog jezika izravno u pregledniku. Solidity programski jezik olakšava implementiranje pametnih ugovora. Remix, uz pisanje, nudi mogućnost prevođenja, testiranja i ispravljanja pogrešaka u kodu, kao i postavljanja samih pametnih ugovora na blockchain. [12]

Node.js je radno okruženje otvorenog koda koje olakšava izvođenje JavaScript koda izvan internetskog preglednika. Izvrstan je kada je potrebno izvršiti ulazno/izlaznu operaciju poput pristupanja bazi podatka. Frontend programerima, uz pisanje koda za klijenta, pruža mogućnost pisanja koda i za server bez potrebe učenja dodatnog jezika. Glavna mu je prednost bogati ekosustav odnosno preko 500 000 biblioteka koje nudi potpuno besplatno [13].

Web3 je skup biblioteka namijenjenih za interakciju sa Ethereum platformom za decentralizirane aplikacije. Kroz ovaj rad će se koristiti za postavljanje ugovora na blockchain kao i za interakciju sa funkcijama pametnog ugovora.

Express je softverski okvir (engl. *Framework*) napisan u JavaScriptu koji se osobito koristi za razvoj serverskog djela web aplikacije. Omogućuje jednostavnu interakciju HTTP zahtjeva i odgovora. U ovom će radu biti korišten za povezivanje implementacije pametnog ugovora i same logike igre.

Ganache-cli je brz i efikasan blockchain simulator kojim se ostvaraju pozivi na blockchain bez troškova pozivanja kroz stvarne čvorove Ethereum sustava. Transakcijski blok se izvršava trenutno, a troškovi transakcija ne postoje i kao takav zapravo predstavlja virtualni osobni blockchain.

4.2. Dizajn igre

Za početak je bilo potrebno osmisliti koncept i dizajn igre zmijica. Unutar HTML (*Hypertext Markup Language*) datoteke definirane su osnovne oznake (engl. *Tag*)(Sl. 4.1) koje će kasnije biti korištene iz JavaScript datoteke za provođenje potrebnih radnji.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link rel="stylesheet" href="style.css" />
    <title>Snake Game</title>
  </head>
  <body>
    <h1>THE SNAKE!</h1>
    <h3>
      Score: <span id="score"></span> Highscore: <span id="highScore"></span>
      <span id="playerName-holder"></span>
    </h3>
    <button id="btnReset">Reset Game</button>
    <canvas id="canvas" width="600" height="600"> </canvas>
    <div class="score">
      <h2 id="scoreEnter"></h2>
      <input type="text" name="playerName" id="playerName" placeholder="Enter your name..."/>
      <button id="submit" type="submit">Submit</button>
    </div>
  </body>
  <script src="script.js"></script>
</html>
```

Slika 4.1. HTML datoteka

Kao glavni element za logiku igre korištena je HTML Canvas oznaka odnosno element za dinamičko grafičko crtanje i to izravno pomoću JavaScripta. Canvas unutar JavaScripta omogućuje metode za crtanja raznih oblika poput krugova, pravokutnika kao i dodavanje slika čime je idealan za crtanje elemenata zmijice. Kako bi se koristile sve mogućnosti koje Canvas pruža, prvo je potrebno dohvatiti taj element iz dokumenta te na njemu pozvati funkciju *getContext* koja vraća kontekst odnosno objekt koji pruža metode za crtanje po elementu. U ovom se slučaju kao parametar funkciji predaje „2d“ kao tip konteksta čime se dobiva objekt koji nudi mogućnost crtanja osnovnih dvodimenzionalnih oblika koji će biti dovoljni za potrebne elemente zmijice.

```
const canvas = document.getElementById("canvas");
const context = canvas.getContext("2d");
const scoreHTML = document.getElementById("score");
const highScoreHTML = document.getElementById("highScore");
const playerNameHTML = document.getElementById("playerName-holder");
document.getElementById("btnReset").addEventListener("click", resetGame);

let boxSize = 30;
let score = 0;
let highScore = 0;
let playerName = "";

let snake;
let direction;
let gameRefreshInterval;

let appleImage = new Image();
appleImage.src = "apple.png";
let apple = setApple();
```

Slika 4.2. Dohvaćanje elemenata iz HTML datoteke i inicijalizacija potrebnih vrijednosti

Veličina Canvas elementa definirana je kao 600 x 600 piksela, a veličina jednog elementa zmijinog tijela je 30 piksela iz čega proizlazi da je veličina aktivnog igrališta zapravo 20 x 20. Na temelju toga definiraju se potrebne varijable kao i slika hrane odnosno jabuke (Sl. 4.2) Koordinate jabuke određuju se pozivanjem *random* funkcije *Math* objekta koja vraća slučajni broj. Glava zmijice pozicionira se na sredinu aktivnog igrališta unutar funkcije *setUpGame*. (Sl. 4.3).

```
function setApple() {
  return {
    x: Math.floor(Math.random() * 20) * boxSize,
    y: Math.floor(Math.random() * 20) * boxSize
  };
}

function setUpGame() {
  document.querySelector(".score").style.display = "none";
  highScoreHTML.innerHTML = highScore;
  playerNameHTML.innerHTML = `(${playerName})`;
  score = 0;
  scoreHTML.innerHTML = score;

  snake = [];
  snake[0] = { x: 10 * boxSize, y: 10 * boxSize };
  direction = "";
  gameRefreshInterval = setInterval(playGame, 100);
}

document.addEventListener("keydown", setDirection);
function setDirection(event) {
  let pressedKey = event.code;

  if (pressedKey == "ArrowLeft" && direction != "RIGHT") direction = "LEFT";
  if (pressedKey == "ArrowRight" && direction != "LEFT") direction = "RIGHT";
  if (pressedKey == "ArrowUp" && direction != "DOWN") direction = "UP";
  if (pressedKey == "ArrowDown" && direction != "UP") direction = "DOWN";
}
```

Slika 4.3. Funkcije za postavljanje početnih stanja i smjera kretanja

Igra se pokreče igračevim pritiskom na bilo koju tipku za smjer čime se indikatoru smjera dodjeljuje vrijednost (Sl. 4.3) na temelju koje će se vršiti određeno pomicanje glave zmijice. Nakon što je vrijednost smjera postavljenja kreće iscrtavanje po Canvas elementu. Kao što je prethodno već navedeno Canvas posjeduje mnoga svojstva i metode, a njihova funkcionalnost dana je tablicom 4.1.

Tablica 4.1. Metode i svojstva Canvas objekta

Metoda / svojstvo	**Opis**
fillStyle	Svojstvo za postavljanje boje, gradijenta ili uzorka za ispunu crteža
fillRect()	Metoda za crtanje ispunjenog kvadrata sa svojstvima zadanim pomoću *fillStyle*. Parametri metode su: x i y koordinate, te širina i visina.
strokeStyle	Svojstvo za postavljanje boje, gradijenta ili uzorka za poteze (linije)
lineWidth	Svojstvo za postavljanje debljine linije
strokeRect()	Metoda za crtanje kvadrata bez ispune sa svojstvima zadanim pomoću *strokeSytle*. Parametri metode su identični kao i za *fillRect* metodu.
drawImage()	Metoda za crtanje slike na Canvas element. Parametri metode su: slika koja se crta, x i y koordinate, te po potrebi širina i visina.

Budući da na Canvasu ostaje zabilježeno sve što je prethodno iscrtano, funkciju *playGame* za crtanje po elementu je potrebno ponovno pozvati svakim pomicanjem zmijice. U ovom slučaju funkcija se poziva svakih 100 milisekundi sve dok igra ne završi. Zmijica se prividno pomiče tako što se pamti posljednji položaj zmijine glave te se funkcijom *pop* izbacuje posljednji element iz niza zmije. Funkcijom *unshift* se zatim na prvo mjesto u niz dodaje novi element sa koordinatama pomaknutim ovisno o smjeru kretanja. Ukoliko se radi o pomicanju lijevo ili desno, mijenja se samo x koordinata glave i to za vrijednost veličine zmijinog tijela, u ovom slučaju to je 30 piksela. Ukoliko koordinate hrane odgovaraju koordinatama glave zmijice iz niza se ne izbacuje element, već se povećava veličina zmijice kao i rezultat igre (Sl.4.4).

```
function playGame() {
  context.fillStyle = "#FFF";
  context.fillRect(0, 0, 600, 600);
  context.strokeStyle = "#FF1D19";
  context.lineWidth = 5;
  context.strokeRect(0, 0, 600, 600);
  for (let i = 0; i < snake.length; i++) {
    context.fillStyle = i == 0 ? "#14A647" : "#FF1D19";
    context.fillRect(snake[i].x, snake[i].y, boxSize, boxSize);
    context.strokeStyle = "#000";
    context.lineWidth = 2;
    context.strokeRect(snake[i].x, snake[i].y, boxSize, boxSize);
  }
  context.drawImage(appleImage, apple.x, apple.y);

  let snakeHead = {
    x: snake[0].x,
    y: snake[0].y
  };
  switch (direction) {
    case "LEFT": snakeHead.x -= boxSize; break;
    case "RIGHT": snakeHead.x += boxSize; break;
    case "UP": snakeHead.y -= boxSize; break;
    case "DOWN": snakeHead.y += boxSize; break;
  }
  if (snakeHead.x == apple.x && snakeHead.y == apple.y) {
    score++;
    scoreHTML.innerHTML = score;
    apple = setApple();
  } else snake.pop();
  let unshiftedHead = {
    x: snakeHead.x,
    y: snakeHead.y
  };
  snake.unshift(unshiftedHead);
  if (collisionCheck(unshiftedHead, snake)) {
    clearInterval(gameRefreshInterval);
    handleScoring();
  }
```

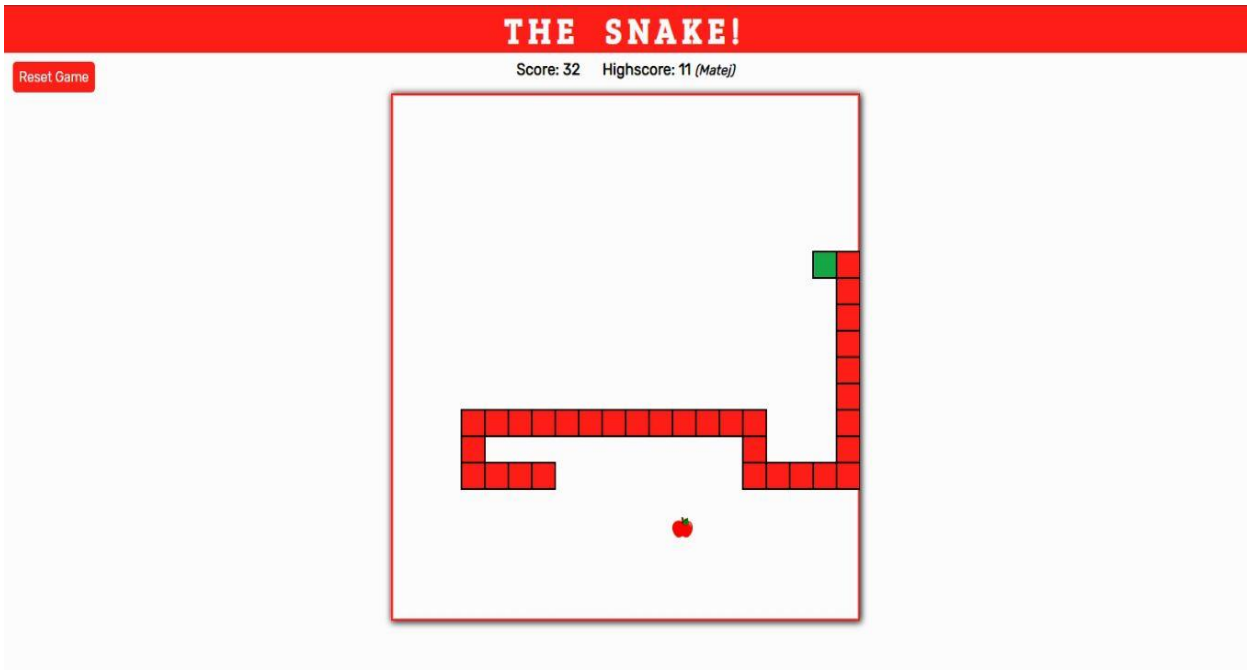
Slika 4.4. *playGame* funkcija za crtanje zmije po Canvas elementu

Na posljetku je još bilo potrebno definirati pravila za koja igra završava. Igra se prekida u slučajevima sudaranja glave zmijice sa vlastitim tijelom ili sudaranja glave sa rubom aktivnog igrališta (Canvas elementa) što provjerava funkcija *collisionCheck* kako je i vidljivo sa slike 4.5.

```
function collisionCheck(head, snake) {
  if (
    head.x < 0 || head.x === 20 * boxSize ||
    head.y < 0 || head.y === 20 * boxSize ) return true;
  for (let i = 1; i < snake.length; i++) {
    if (head.x == snake[i].x && head.y == snake[i].y) return true;
  }
  return false;
}
```

Slika 4.5. Metoda za provjeru istinitosti uvjeta za završetak igre

Kako bi oznake HTML datoteke poprimili željeni izgled i položaj na samoj stranici potrebno je definirati stil tih elemenata pomoću CSS (*Cascading Style Sheet*) opisnog jezika. Nakon testiranja osnovnog rada igre, za korištenje sustava bodovanja bilo je potrebno implementirati pametni ugovor.



Slika 4.6. Korisničko sučelje igre

4.3. Pametni ugovori

Klasičan se ugovor definira između dviju ili više strana kao očitovanje suglasnosti između tih subjekata. Problem nastaje jer se u taj sklopljeni ugovor uključuje treća strana (osoba ili institucija). Pametni ugovori definiraju identičan odnos kao i klasični ugovori uz bitnu razliku što su prevedeni u računalni kod, lišeni treće strane i potpuno decentralizirani zahvaljujući blockchainu. Upotrebom pametnih ugovora dobivamo podatak sa kojim nije moguće manipulirati. Začetnik ideje pametnih ugovora bio je računalni stručnjak i kriptograf Nick Szabo nakon što je uvidio da se decentralizirane poslovne knjige (engl. *Ledger*) mogu upotrebljavati upravo kao pametni ugovori.[16] Temelji se na logici da ukoliko se dogodi jedan događaj, onda se izvršava i određena radnja. Jednom postavljen kod pametnog ugovora ne može se promijeniti.

Pametni ugovori mogu imati vrlo široku primjenu kroz decentralizirane aplikacije. Banke ga mogu primjenjivati za izdavanje kredita ili automatiziranu isplatu. Javnosti može biti od koristi kroz mogućnost spremanja sigurnosnih podataka građana, potvrđivanje njihova identiteta i

automatizacije procesa glasanja, plaćanja, investiranja itd. Svojom autonomnošću, sigurnošću, brzinom i učinkovitošću praktički nemaju granice upotrebe.

Ethereum sustav je javna blockchain platforma za izvršavanje i pohranu pametnih ugovora. Ethereum za svoj rad koristi kriptovalutu Ether (skraćeno ETH). Bitcoin mreža razumije samo jednostavne naredbe za rad sa novcem dok su mogućnosti koje pruža Ethereum daleko od jednostavnih naredbi za isključivi rad s novcem. Uz Ether kao kriptovalutu Ethereum se oslanja i na token zvan *gas*. Upravo su njih dvoje osiguravatelji optimiziranog i efikasnog koda. Izvršavanje akcija unutar sustava troši resurse i zbog toka se koristi *gas* koji označava težinu izvršavanja akcije. Na primjer, postavljanje ugovora na Ethereum platformu je puno složeniji proces od pozivanja pojedine instrukcije pa će samim time zahtijevati veću količinu *gasa*. Također količina *gasa* potrebnog za akciju se ne mijenja dok vrijednost valute Ether može varirati. Slanjem zahtjeva za promjenu stanja ugovora kreira se i nova transakcija. Svakom novom transakcijom dodaje se novi blok podataka što je vidljivo na slici 4.12.

4.4. Implementacija pametnog ugovora

4.4.1. Solidity

Solidity je objektno-orijentirani jezik namijenjen za implementaciju pametnih ugovora na razne blockchain platforme, a ponajprije na Ethereum. Na Solidity su izravno utjecali C++, Python te JavaScript što se ponajviše može vidjeti kroz samo pisanje ugovora. Osim što omogućuje nasljeđivanje i brojne biblioteke, moguće je i stvoriti ugovore za glasanje, aukcije, grupno financiranje (engl. *Crowdfunding*) i još mnoge druge mogućnosti [17].

Pisanje samih ugovora poprilično sliči pisanju klasa u drugim objektno-orijentiranim jezicima, no za početak je potrebno definirati verziju prevoditelja Solidity koda(Sl. 4.7.). Pragma definira dodatne mogućnosti i provjere prevoditelja te se definiranjem točne verzije prevoditelja osigurava da će se kod uvijek izvršavati onako kako je i zamišljeno.

Za bodove je definirana varijabla tipa uint (*unsigned integer*) jer bodovi mogu biti isključivo pozitivne vrijednosti. Nadalje su definirane dvije metode za dohvaćanje vrijednosti varijabla. Ključna riječ *View* uz metodu označava da funkcija ne mijenja stanje i na taj način omogućuje poziv istih kroz *call* funkciju web3 objekta. Također definirana je javna metoda za postavljanje novog stanja ugovora koja kao parametre prima bodove i ime igrača te se još dodatno provjerava istinitost tvrdnje da je pristigli broj bodova veći od dosadašnjeg broja bodova. Ako se

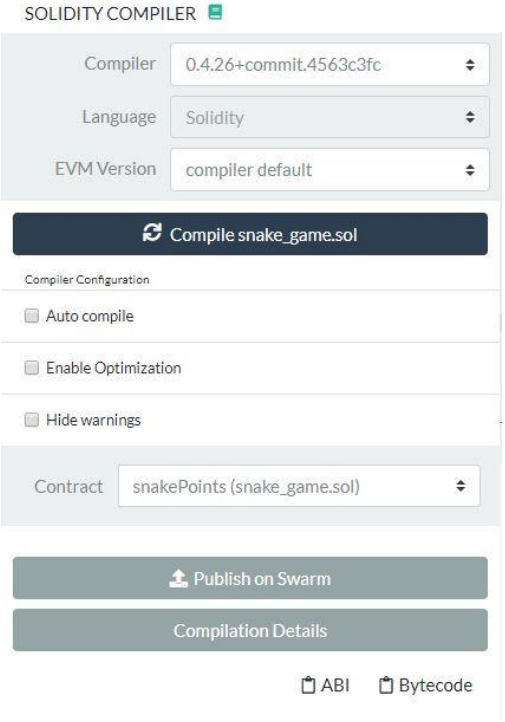
broj bodova pokaže manji od trenutnog, poziva se ugrađena metoda *revert* koja opovrgava trenutni poziv, ne mijenja stanje i vrača korisniku potrošenu količinu *gasa*. Metoda za postavljanje parametara će se pozivati preko *send* funkcije web3 objekta zbog toga što se kao rezultat pozivanja metode mijenjaju uvjeti odnosno stanje ugovora.

Home snake_game.sol

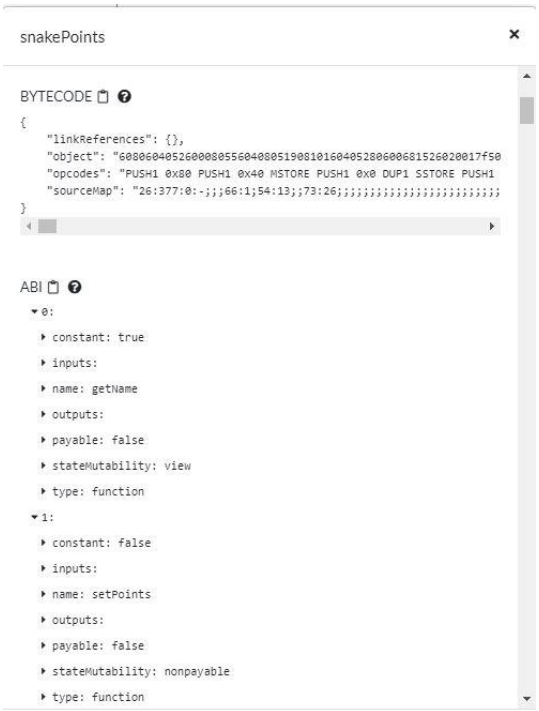
```
1  pragma solidity ^0.4.26;
2
3  contract snakePoints {
4
5      uint points=0;
6      string playerName="Player";
7
8      function getPoints() public view returns(uint){
9          return points;
10     }
11     function getName() public view returns (string){
12         return playerName;
13     }
14     function setPoints(uint _points, string _playerName) public {
15         if(_points>points){
16             points=_points;
17             playerName=_playerName;
18         }
19         else revert();
20     }
21 }
```

Slika 4.7. Solidity kod pametnog ugovora *snakePoints*

Nakon što je ugovor *snakePoints* napisan, kod je potrebno prevesti i za to se koristi već ranije zadana verzija prevoditelja (Sl. 4.8.). Rezultat prevođenja biti će ABI (*Application Binary Interface*) i heksadecimalni oblik prevedenog koda (engl. *Bytecode*) (Sl. 4.9) koji će se izvršavati na Ethereum platformi. ABI sučelje služi za opisivanje funkcija pametnog ugovora i interakciju sa metodama pametnog ugovora. Zahvaljujući njegovim podacima kao i podacima Bytecoda ugovor se može postaviti na Ethereum blockchain platformu preko Web3 biblioteke.



Slika 4.8. Remix sučelje za prevođenje Solidity programskog koda



Slika 4.9. Prikaz byteCoda i ABI sučelja prevedenog koda

4.4.2. Web3

Prije bilokakve upotrebe pametnog ugovora, najprije ga je potrebno postaviti na Ethereum blockchain platformu. Za implementaciju ugovora biti će korišteni već ranije navedeni Web3, Ganache-cli i Express.js. Za njihovu instalaciju koristi se Node.js. Upisivanjem naredbe *npm -i web3 ganache-cli express* unutar naredbenog retka Node.jsa, sve tri potrebne biblioteke se instaliraju u željenu mapu. Kako bi ugovor bio ispravno postavljen potrebno mu je predati ABI sučelje i Bytecode koji su ranije dobiveni prevođenjem Solidity koda. Također je potrebna i adresa odnosno račun sa kojeg će se taj ugovor i postaviti, a taj račun omogućuje simulator Ganache-cli (Sl. 4.10.). Za ovu simulaciju korištena je prva dostupna adresa. Ganache-cli također ograničava maksimalnu dopuštenu količinu *gasa* kao i maksimalnu količinu *gasa* po transakciji. Nakon što su svi podaci potrebni za postavljanje ugovora osigurani stvara se novi objekt *Contract* (Sl. 4.11.). Na tom objektu poziva se metoda *deploy* koja će nakon uspješnog postavljanja vratiti novu instancu ugovoru koju će se zapravo poslati na Etherum funkcijom *send*. Toj se funkciji kao parametri predaju adresa sa koje se šalje kao i maksimalna dopuštena količina *gasa* koja, ukoliko se pređe, sprječava izvršavanje akcije i samim time osiguranje od dodatnih troškova.

```
Matej@HP-Pavilion MINGW64 ~
$ ganache-cli
Ganache CLI v6.4.5 (ganache-core: 2.5.7)

Available Accounts
==================
(0) 0x7b82273d12e389827efc69c5dac35ca21be3ca59 (~100 ETH)
(1) 0x86239176ba13598c58232fc3a287b9e3c78d061f (~100 ETH)
(2) 0x24ff081115147c2c555b9726a30c4fc4940174eb (~100 ETH)
(3) 0xc4a07edc0f2d6e626b2c6c18da76b6f61b913be9 (~100 ETH)
(4) 0x1ff6ef2f7214a94fd1039d8881fab331d9c62c9e (~100 ETH)
(5) 0x43427e54ac96edfd8c4c37d2c09e82ca9b3694e2 (~100 ETH)
(6) 0x43dad5a26b35885846e7c9ee4e70e0b212038bf7 (~100 ETH)
(7) 0xfabe95020b0d4d17acf393e540dc76776abba437 (~100 ETH)
(8) 0x8c5858ae57070ffddda3458a69e9e3171817d726 (~100 ETH)
(9) 0x56adfd8ce865d931755f68a0cf4ee3a4827c9e88 (~100 ETH)

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Listening on 127.0.0.1:8545
```

Slika 4.10. Ganache-cli dostupne adrese za postavljanje ugovora

```
const Web3 = require("web3");
const web3 = new Web3("http://127.0.0.1:8545");

const ABI = […
const byteCode =
  "6080604052600080556040805190810160405280600681526020017f506c61796572000
const address = "0x7b82273d12e389827efc69c5dac35ca21be3ca59";

var contract = new web3.eth.Contract(ABI);

contract
  .deploy({ data: byteCode })
  .send({ from: address, gas: 4700000 }, (error, transactionHash) =>
    console.log("Error:", error, transactionHash)
  )
  .then(function(result) {
    console.log(result);
  });
```

Slika 4.11. Programski kod za postavljanje ugovora na blockchain

```
eth_gasPrice
eth_sendTransaction

  Transaction: 0xbd9d70ce4a149aac23b8adb9ad1490e85b65df76f8989298ea1ab638a80afe9a
  Contract created: 0x369cb11291bb9c9e56ca062caa08112879cdd7ab
  Gas usage: 307759
  Block Number: 1
  Block Time: Sun Jul 21 2019 17:42:51 GMT+0200 (GMT+02:00)

eth_getTransactionReceipt
eth_getCode
```

Slika 4.12. Rezultat postavljanja pametnog ugovora na blockchain pomoću Ganache-cli simulatora

Da je ugovor uspješno postavljen može se vidjeti iz Ganache-cli simulatora gdje kao rezultat izvršavanja dobivamo detalje transakcije. Ugovor je postavljen kao prvi blok u nizu. Nakon što je ugovor uspješno postavljen na slobodnu adresu, tom ugovoru možemo pristupiti isključivo poznavajući tu jedinstvenu adresu koja je vidljiva sa slike 4.12. Nakon što je stvoren novi objekt *Contract* sa adresom na kojoj se nalazi postavljeni ugovor (Sl. 4.13), na tom se objektu konačno mogu pozivati metode definirane u Solidity jeziku i to zahvaljujući generiranom ABI sučelju. Metode koje vraćaju rezultat je potrebno pozvati funkcijom *call* koja će pozvati metodu pametnog ugovora, međutim bez slanja transakcije i samim time mreža nije informirana o akciji pa će i stanje valute ostati ne promijenjeno. Za razliku od *call*, *send* funkcija će biti korištena kada je potrebno promijeniti stanje ugovora odnosno kada je potrebno izvršiti transakciju i metodu pametnog ugovora. Kao što je već spomenuto, za poziv svake akcije koja se odvija na Ethereum platformi potrebna je određena količina *gasa*. Samim time metoda *send* svakim svojim pozivom te zatim kreiranjem i slanjem transakcije troši *gas* i u konačnici valutu Ether (ETH).

4.4.3. Povezivanje sa igrom

Ugovor je kreiran, postavljen i spreman za korištenje, međutim dva dijela igre su potpuno ne povezana. Za povezivanje će biti korišten Express softverski okvir. Za početak je potrebno pozvati Express biblioteku te *express* funkciju za stvaranje aplikacije (Sl. 4.13). Sva funkcionalnost igre postavlja se u mapu *public* i zahvaljujući Expressu ta mapa može biti postavljena kao statična, što znači da će se moći pokrenuti zajedno sa svim njenim datotekama (HTML, CSS i JavaScript). Pozivom funkcije *listen*, koja vraća instancu HTTP servera, igra se sa svim datotekama pokreće na lokalnom serveru.

```
const Web3 = require("web3");
const web3 = new Web3("http://127.0.0.1:8545");

const address = "0x7b82273d12e389827efc69c5dac35ca21be3ca59";
const contractAddress = "0x369cb11291bb9c9e56ca062caa08112879cdd7ab";
const ABI = [···

var contract = new web3.eth.Contract(ABI, contractAddress);

const express = require("express");
const app = express();

app.use(express.static("public"));

const PORT = 3000;
app.listen(PORT, () => console.log(`SERVER STARTED ON PORT: ${PORT}`));
```

Slika 4.13. Programski kod za dohvaćanje postavljenog ugovora i pokretanja lokalnog servera

Express je od posebne koristi jer omogućuje slanje i primanje HTTP zahtjeva. *Get* funkcija je korištena za postavljanje rute (putanje) koja jednom kada se pozove vraća odgovor (engl. *Response*) kojeg je dobila od metode pametnog ugovora. Dobiveni odgovor će funkcijom *send* biti poslan natrag u preglednik. Na identičan način će biti pozvana i *post* funkcija, ali uz bitnu prednost što je sigurnija, a to je ono što je potrebno za slanje osjetljivih zahtjeva.

```
app.post("/add-highscore", (request, response) => {
  const points = request.query.points;
  const name = request.query.name;
  console.log("Player: ", name, "sent: ", points);
  contract.methods.setPoints(points, name)
    .send({ from: address })
    .then(e => response.send("Points added"));
});
app.get("/get-highscore", (request, response) => {
  contract.methods.getPoints()
    .call({ from: address })
    .then(result => response.send(result));
});
app.get("/get-playerName", (request, response) => {
  contract.methods.getName()
    .call({ from: address })
    .then(result => response.send(result));
});
```

Slika 4.14. Putanje za slanje zahtjeva i pozivanje metoda pametnog ugovora

Postavljenjem putanja stvoreni su uvjeti za poziv metoda pametnog ugovora u trenutcima kada igra to zahtjeva. Podaci se iz blockchaina dohvaćaju odmah prilikom učitavanja igre metodom *fetch*, koja dobiva odgovor u obliku bodova i imena.(Sl. 4.15) Dobiveni podaci se spremaju u varijable koje se koriste za prikaz podataka u HTML datoteci. Nakon dohvaćanja podataka korisnik može započeti sa igrom.

```
getHighScore();
function getHighScore() {
  fetch("/get-highscore").then(res => res.json())
  .then(data => {
      highScore = data;
      fetch("/get-playerName").then(res => res.text())
        .then(data => {
          playerName = data;
          setUpGame();
        });
    });
}
```

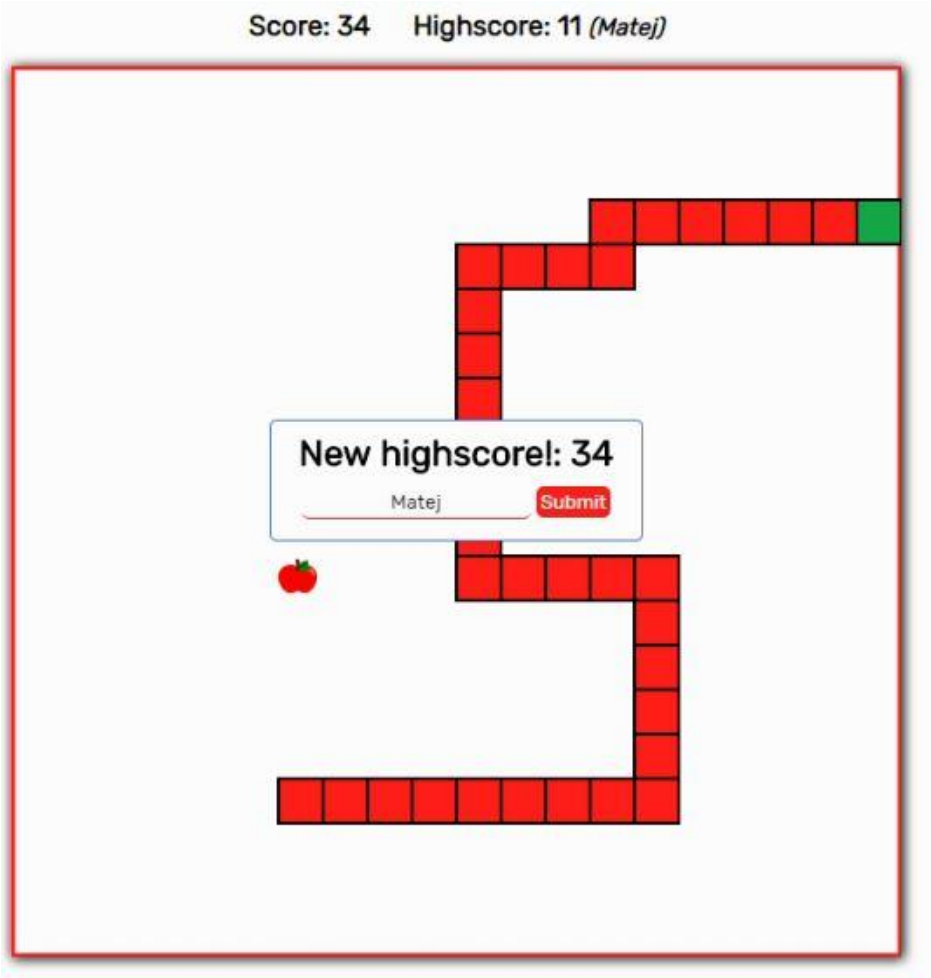
Slika 4.15. Dohvaćanje podataka iz blockchain mreže

Slanje POST zahtjeva se pruža tek kada je korisnik ostvario veći rezultat od onog dobivenog iz blockchaina. Korisnik klikom na „submit“ gumb šalje zahtjev za slanjem novog bodovnog stanja na blockchain. (Sl.4.16) Kada je akcija provedena, u Ganache-cli simulatoru ćemo dobiti ispis *hash* vrijednosti transakcije, količinu *gasa* utrošenog za provođenje te akcije, redni broj postavljenog bloka i vrijeme izvršavanja akcije što je i vidljivo sa slike 4.18. Korisnik u konačnici uopće ne mora biti svjestan da se podatak dohvaća i da se transakcije vrše na blockchainu, a u isto je vrijeme osigurano da se jednom ubačeni broj bodova i ime igrača ne mogu ne ovlašteno mijenjati bez poziva metode za slanje transakcije. Nakon što je igra završila, korisnik pritiskom na „reset“ gumb vrača sve podatke na početne vrijednosti. Jedini podatci koji se ne vračaju na početne vrijednosti već dohvaćaju sa blockchaina su rezultat i ime igrača sa najboljim rezultatom.

```
function handleScoring() {
  let btnSubmit = document.getElementById("submit");
  document.querySelector(".score").style.display = "block";
  document.getElementById("scoreEnter").innerHTML =
    score <= highScore ? `GAME OVER!<br>Your score: ${score}` : `New highscore!: ${score}`;
  score <= highScore
    ? ((document.getElementById("playerName").style.display = "none"),
      (btnSubmit.style.display = "none"))
    : ((document.getElementById("playerName").style.display = "inline"),
      (btnSubmit.style.display = "inline"));
  btnSubmit.addEventListener("click", submitScore);
}

function submitScore() {
  document.getElementById("submit").removeEventListener("click", submitScore);
  let name = document.getElementById("playerName").value;
  console.log(score, name);
  if (score > highScore) {
    var xhr = new XMLHttpRequest();
    xhr.open("POST", `/add-highscore?points=${score}&name=${name}`);
    xhr.send();
  }
}
```

Slika 4.16. Kod za slanje HTTP zahtjeva za upis najboljeg rezultata na blockchain



Slika 4.17. Korisnički prikaz za slanje novog najboljeg rezultata

```
eth_gasPrice
eth_sendTransaction

  Transaction: 0x131ee0b2439c1bac12e31f4f1906da0f94ac526642d4ac3015d9848e9aad1bbe
  Gas usage: 53325
  Block Number: 2
  Block Time: Sun Jul 21 2019 17:52:02 GMT+0200 (GMT+02:00)

eth_getTransactionReceipt
eth_call
eth_call
eth_gasPrice
eth_sendTransaction

  Transaction: 0xc4ea60007b04b3fe25b14cf2ad12992bccba7f3905129a133a6acb02187bdd8f
  Gas usage: 38325
  Block Number: 3
  Block Time: Sun Jul 21 2019 17:53:44 GMT+0200 (GMT+02:00)

eth_getTransactionReceipt
```

Slika 4.18. Ganache-cli rezultat dobiven provođenjem akcije za slanje zahtjeva za upis najboljeg rezultata na blockchain

5. ZAKLJUČAK

Kroz sve navedene primjere igara može se zapravo zaključiti kako je industrija video igara koje koriste blockchain tek u svojim začetcima. Među ključnim problemima se nameće nedostatak infrastrukture za što ne treba otići dalje od igre *CriptoKitties* i platforme Ethereum. Ethereum jednostavno nije bila spremna za pomamu koja je nastala za igrom što je u konačnici rezultiralo povećanjem naknada potrebnih za provođenje transakcija, no rješenje možda nudi EOS.IO. To je platforma koja omogućuje puno veću obradu transakcija zbog svoga *proof-of-stake* algoritma. Te dvije platforme su trenutno vodeće po broju decentraliziranih aplikacija i blockchain igara koje se nalaze na njima. Možda je rješenje problema upravo kombinacija dviju platformi i mogućnosti koje one pružaju.

Iako je u začetcima kroz ovaj se završni rad vidjelo kako su se svi događaji ključni za daljnji razvoj počeli odvijati unazad par godina. Tehnologija blockchain igara napreduje sve brže i tvrtke uključene u proces su tek počele otkrivati koje sve sfere mogu biti unaprijeđene kroz blockchain. O mogućnostima i prednostima blockchaina u industriji video igara govori i to kako se sve više divova industrije, poput Ubisofta, uključuju u istraživanja ovog područja. Iz Sonya dolaze najave kako bi se za spremanje podataka o vlasništvu mogli uskoro prebaciti na blockchain. Također Epic Games, tvorci igre *Fortnite*, zainteresirani su za istraživanje ovog područja.

Kroz primjer JavaScript igre zmijica se prikazalo na koji se način pomoću Solidtiy programskog jezika i simulacije na Ethereum platformi može iskoristiti blockchain kako bi se spremili podaci igrača koji ostvari najbolji rezultat i osigurati povijest i sigurnost tih podataka. U vremenu kada igračevi uspjesi u igri i ostali njegovi podaci postaju sve važniji, kada eSport generira ogromne količine novca i kada prevaranti na sve moguće načine pokušavaju doći do računa drugih igrača, blockchain se nameće kao tehnologija koja bi mogla izmijeniti i unaprijediti sva ta područja.

LITERATURA

[1] CryptoKitties Guide [online], cryptokitties.co, dostupno na: https://guide.cryptokitties.co/guide/ [29.7.2019.]

[2] CryptoKities Source code [online], etherscan.io, dostupno na: https://etherscan.io/address/0x06012c8cf97bead5deae237070f9587f8e7a266d#code [29.7.2019.]

[3] P.Bock, Genesis Cat, the #1 Cryptokitten, sells for 250 ETH [online], steemit.com. dostupno na: https://steemit.com/cryptokitties/@pbock/genesis-cat-the-1-cryptokitten-sells-for-250-eth [29.7.2019]

[4] N.Varshney, Someone paid $170,000 for the most expensive CryptoKitty ever [online], thenextweb.com, dostupno na: https://thenextweb.com/hardfork/2018/09/05/most-expensive-cryptokitty/ [29.7.2019.]

[5] Dapp Raddar [online], dappradar.com, dostupno na: https://dappradar.com/rankings/category/games [30.7.2019.]

[6] Prospectors [online], prospectors.ios, dostupno na: https://prospectors.io/about_game?locale=en [30.7.2019.]

[7] DreamTeam [online], dreamteam.gg, dostupno na: https://dreamteam.gg [30.7.2019.]

[8] C. Stead, Ubisoft reveals HashCraft, a secret new blockchain game [online], finder.com.au, dostupno na: https://www.finder.com.au/hashcraft-ubisoft-blockchain-game [31.7.2019.]

[9] CryptoZombies [online], cryptozombies.io, dostupno na: https://cryptozombies.io/ [31.7.2019.]

[10] Blockchain Game Alliance [online], blockchaingamealliance.org, dostupno na: http://blockchaingamealliance.org [31.7.2019.]

[11] Visual studio Code [online], Microsoft Corporation, dostupno na: https://code.visualstudio.com/docs [18.7.2019.]

[12] Welcome to Remix documentation! [online], Remix, dostupno na: https://remix-ide.readthedocs.io/en/latest/ [18.7.2019.]

[13] Introduction to Node.js [online], Node.js Foundation, dostupno na: https://nodejs.dev/ [18.7.2019.]

[14] Web3 [online], Ethereum, dostupno na: https://web3js.readthedocs.io/en/v1.2.0/web3.html [18.7.2019.]

[15] Ganache CLI [online], Nathereum.com, dostupno na: https://docs.nethereum.com/en/latest/ethereum-and-clients/ganache-cli/ [18.7.2019.]

[16] A. Rosic, Smart Contracts: The Blockchain Technology That Will Replace Lawyers [online], BlockGeeks.com, dostupno na: https://blockgeeks.com/guides/smart-contracts/ [21.7.2019.]

[17] Solidity [online], Ethereum, dostupno na: https://solidity.readthedocs.io/en/v0.4.25/ [21.7.2019.]

SAŽETAK

Kroz ovaj je rad opisano trenutno stanje industrije video igara koje koriste blockchain. Opisane su trenutno najpopularnije blockchain igre kao što su: CryptoKitties, EOS Knights i Prospectors. Također, navedene su ideje kako se ovo područje može dalje razvijati i koje bi bile prednosti korištenja blockchaina u ovom segmentu. Opisane su trenutno najpopularnije platforme koje se koriste pri izradi blockchain igara kao i koncept pametnih ugovora. U okviru rada napravljena je i JavaScript igra zmijica. Podaci najboljeg rezultata ostvarenog u igri se spremaju na blockchain. Ti podaci ostaju zauvijek zapisani na blockchain i gotovo je nemoguće manipulirati sa njima.

Ključne riječi: Blockchain, Ethereum, Pametni ugovori, Video igre

ABSTRACT

Blockchain and Video games

This final assignment describes the current state of the blockchain gaming industry. It describes currently the most popular blockchain games such as: CryptoKitties, EOS Knights and Prospectors. Furthermore, ideas for the further development in this area has also been described as well as the benefits of using the blockchain within the video games. This paper also describes most popular blockchain gaming platforms as well as the smart contracts. Within this work, a Snake game was made using the JavaScript language. Data of the best result achieved in the game is stored on blockchain. This data stays on the blockchain permanently and it's almost impossible to manipulate with it.

Keywords: Blockchain, Ethereum, Smart Contracts, Video games

ŽIVOTOPIS

Matej Posavac je rođen 27. lipnja 1996. godine u Osijeku. Završio je Osnovnu školu Ivana Filipovića u Osijeku. 2011. godine upisuje Elektrotehničku i prometnu školu u Osijeku koju završava 2015. godine i te iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Ima iskustvo rada kao tehnička podrška u Hrvatskom Telekomu.

Potpis:

PRILOZI

[1] Izvorni kod igre zmijica