Osnovna šola Velika Nedelja

Naša šola v virtualnem svetu iz kock

Računalništvo ali telekomunikacije

raziskovalna naloga

Avtor: Aljaž Strnad

Mentor: Božidar Muršec

Somentor: Boštjan Strnad

Velika Nedelja, 2021

ZAHVALA

Spoštovanima mentorjema Božidarju Muršcu in Boštjanu Strnadu se iskreno zahvaljujem za vso pomoč in podporo pri pripravi raziskovalne naloge. Hvala tudi g. Radovanu Krajncu za vse koristne informacije pri snovanju raziskave in lektorici ge. Lani Meglič Kovačič.

KAZALO

KAZALO SLIK

KAZALO TABEL

POVZETEK

Odločil sem se raziskati, ali je v Minecraftu možno zgraditi model realne zgradbe, katera znanja ter spretnosti pri tem potrebujem in ali sem ta znanja pridobil v času pouka rednih predmetov.

Minecraft je priljubljena konstrukcijska igra, v kateri je vse sestavljeno iz kock. To igro sem poznal že prej, od letos pa nam je učencem na voljo tudi izobraževalna različica omenjene igre.

Posebnost te različice (Minecraft Education) je t. i. agent. Vloga agenta je, da pomaga pri izgradnji različnih gradbenih elementov in stavb. Igralec pripravi (sprogramira) niz ukazov za agenta, ki s pomočjo teh ukazov gradi zahtevane oblike. Sestavljanje programa lahko poteka v vizualni blokovni (MakeCode) ali tekstovni obliki (JavaScript ali Python).

Odločil sem se, da v Minecraftu zgradim našo osnovno šolo Velika Nedelja. Pri tem sem naletel na veliko izzivov, ki sem jih moral sproti reševati, ob tem pa sem opazoval, katere elemente računalniškega mišljenja razvijam pri reševanju teh problemov. Za doseganje ciljev je bilo potrebnih veliko korakov: merjenje in priprava tlorisov šole, programiranje gradnje šole v virtualnem svetu po fazah ter reševanje sprotnih manjših težav (npr. oblika in velikost gradbenih elementov). Skozi raziskavo sem spoznaval programski jezik JavaScript in hkrati razvijal veščine in spretnosti, ki so v današnjem času zelo pomembne, med njimi še prav posebej izpostavljam računalniško mišljenje in njegove koncepte.

Ključne besede: računalniško mišljenje, koncepti računalniškega mišljenja, virtualni model, Minecraft, programiranje

ABSTRACT

I decided to research whether it is possible to build a model of a real building in Minecraft, what knowledge and skills I need, and whether I acquired this knowledge during the lessons of regular subjects.

Minecraft is a popular construction game in which everything is made up of cubes. I knew this game before, and as of this year, an educational version of the game is also available to our students.

The peculiarity of this version (Minecraft Education) is the so-called agent. The role of the agent is to assist in the construction of various building elements and buildings. The player prepares (programs) a set of commands for the agent, which uses these commands to build the required shapes. The program can be compiled in visual block (MakeCode) or text format (JavaScript or Python).

I decided to build our elementary school Velika Nedelja in Minecraft. In doing so, I encountered many challenges that I had to solve, while observing what elements of computational thinking I was developing in solving these problems. Many steps were needed to achieve the goals: measuring and preparing school floor plans, programming school construction in the virtual world in phases, and solving ongoing minor problems (e.g., the shape and size of building elements). Through the research, I learned about the JavaScript programming language, and at the same time developed skills and abilities that are very important nowadays, among which I especially emphasize computational thinking and its concepts.

Keywords: computational thinking, computational thinking concepts, virtual model, Minecraft, programming

1 UVOD

S programom Minecraft imam precej izkušenj. Gre za igro, v kateri je cel svet sestavljen iz kock. Izkušnje sem si nabral predvsem z gradnjo velikih objektov, kot so nogometni stadioni. Pred kratkim sem odkril posebno različico Minecrafta – Minecraft Education Edition. V tej različici lahko sestaviš program tako, da agent gradi različne konstrukcije. Dobil sem idejo, da bi bilo zanimivo zgraditi kakšno realno stavbo. Ker pa je veliko svetovno znanih stavb že zgrajenih v Minecraftu, sem se osredotočil na znane okoliške zgradbe. Odločil sem se, da bom zgradil šolo, ki jo tudi sam obiskujem. Izgradnja naše šole mi je bila v velik izziv, ker je zgradba zelo kompleksna.

V današnjem sodobnem času postajajo računalniška znanja in digitalne kompetence vse bolj pomembne. Izdelava virtualnega modela šole predstavlja problem, ki ga je mogoče rešiti le s pomočjo uporabe omenjenih znanj in veščin. Pojavi se vprašanje, ali ta znanja pridobimo v času pouka v osnovni šoli (izbirnih predmetov, interesnih dejavnosti in drugih neobveznih oblik sem ne moremo šteti, saj jih ne obiskujejo vsi učenci).

Raziskati sem želel, ali je mogoče z znanji, pridobljenimi pri obveznih predmetih v osnovni šoli, rešiti problem s pomočjo računalniškega mišljenja in programiranja, npr. prenesti model zgradbe iz realnega sveta v obliko prilagojenega modela v virtualnem svetu.

Šolska zgradba je sestavljena iz starega dela (zgrajen v 18. stoletju) in novega dela (zgrajen leta 1984). Zanimalo me je tudi, kako izvesti posamezne postopke, ki so potrebni za končno izgradnjo velike stavbe v Minecraftu, od iskanja ali merjenja in risanja načrtov do sestavljanja kode ter iskanja rešitev za izzive, ki se bodo pojavljali na poti do cilja.

1.1 Raziskovalna vprašanja

1. Katere postopke bom moral izvesti, preden bom lahko začel graditi šolsko stavbo v Minecraftu?
2. Ali imam dovolj znanj, ki sem jih pridobil v času rednega pouka, da izvedem vse postopke?
3. Katera potrebna znanja že imam in kje v času rednega pouka sem jih pridobil?
4. Katera znanja moram še pridobiti, da lahko dokončam virtualni model šole?
5. Ali bo mogoče celoten program za gradnjo virtualnega modela šole sestaviti z blokovnim programiranjem?
6. Kje vse se bodo pojavljali vzorci pri gradnji virtualnega modela šole?
7. Ali obstaja več različnih programskih rešitev za izgradnjo določenega dela virtualnega modela šolske zgradbe?

1.2 Hipoteze

1. Izgradnjo šolske stavbe v Minecraftu bom lahko izvedel na osnovi načrtov šolske zgradbe.
2. V času rednega pouka nisem pridobil dovolj znanj za gradnjo virtualnega modela šole.
3. V času rednega pouka sem pridobil določene digitalne kompetence pri uporabi računalnika ter v manjšem obsegu tudi koncepte računalniškega mišljenja pri posameznih rednih predmetih.
4. Manjkajo mi predvsem znanja s področja računalniškega programiranja.
5. Gradnjo celotne šole bo mogoče sprogramirati zgolj s pomočjo blokov.
6. Nekatera stopnišča v šoli so sestavljena iz enakega števila stopnic z enako višino, zato predstavljajo ponavljajoči se vzorec.
7. Pri gradnji posameznega dela šole bo obstajala zgolj ena programska rešitev, če zanemarimo vrstni red korakov.

2 TEORETIČNI DEL

2.1 Računalniško mišljenje

2.1.1 Definicija računalniškega mišljenja

Pojem računalniško mišljenje je prvi uporabil Seymour Papert v knjigi Mindstorms (1980). V knjigi je opisoval možnosti uporabe računalnikov v šoli pri reševanju problemov in drugačnem načinu razmišljanja. Avtor je v knjigi zapisal, da je resnično računalniško pismena tista oseba, ki zna ne samo uporabiti računalnik za rešitev problema, temveč zna tudi oceniti, kdaj je vključevanje računalnika v reševanje problemov smiselno (Kranjc, Košir in Čotar Konrad, 2017).

International Society for Technology in Education in Computer ScienceTeacher Association so leta 2011 predstavili definicijo računalniškega mišljenja za potrebe izobraževanja otrok, starih med petimi in osemnajstimi leti (ISTE in CSTA, 2011).

Navajajo, da je računalniško mišljenje način reševanja problemov, ki vključuje (vendar ni omejeno na) naslednje značilnosti:

- oblikovanje problemov na način, ki nam omogoča uporabo računalnika in drugih orodij za njihovo reševanje;
- logično organiziranje in analiziranje podatkov;
- predstavljanje podatkov z abstrakcijami, kot so modeli in simulacije;
- avtomatizacija rešitev z algoritmičnim razmišljanjem (vrsta zaporednih korakov);
- prepoznavanje, analiziranje in izvajanje možnih rešitev s ciljem doseganja najučinkovitejše kombinacije korakov in virov;
- posploševanje in prenos tega postopka reševanja problemov na najrazličnejše težave.

Izraz računalniško mišljenje je predstavila tudi Jeanette M. Wing leta 2006. Wingova trdi, da »računalniško mišljenje temelji na moči in mejah računalniškega procesa, ne glede na to, ali jih izvaja človek ali stroj«. Razpravlja tudi o potrebi po združitvi računalniške in človeške inteligence. Na splošno povzema, da "razmišljanje kot računalničar pomeni več kot le 'programiranje'. Zahteva razmišljanje na več ravneh abstrakcije" (Wing, 2006).

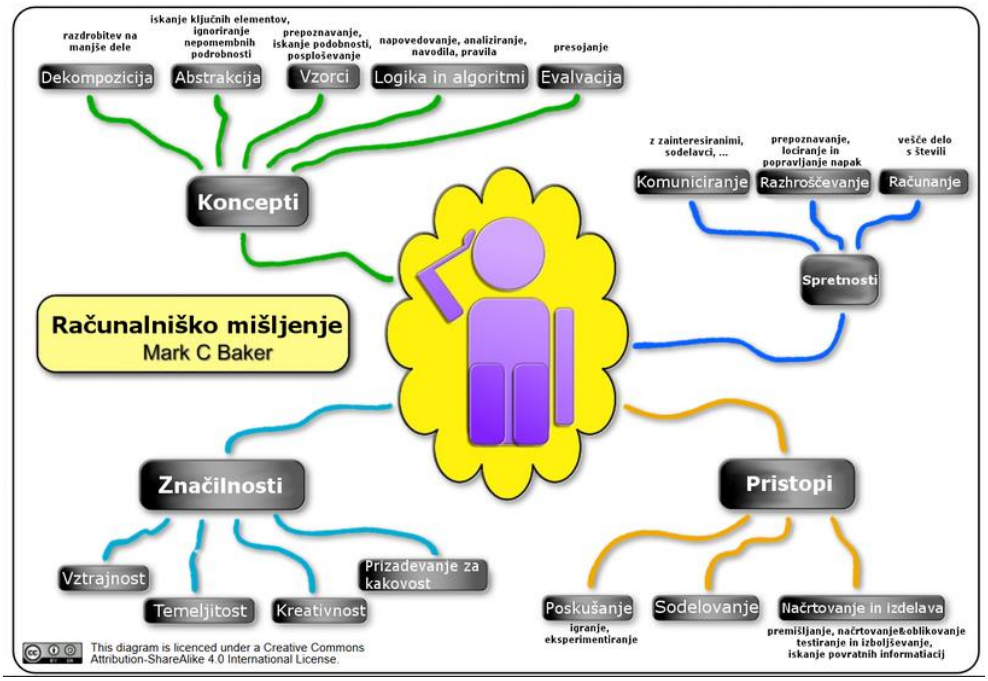
Wingova izpostavlja najpogostejša "napačna prepričanja" o tem, kaj so vsebine računalniškega mišljenja (Kranjc, Košir in Čotar Konrad, 2017).

Tabela 1: Vsebine računalniškega mišljenja (Kranjc, Košir in Čotar Konrad, 2017)

Kaj je računalniško mišljenje?	**Kaj ni računalniško mišljenje?**
Konceptualizacija: računalniško mišljenje presega običajen nivo računalniškega programiranja; pomeni razmišljanje, ki izhaja iz algoritemskega mišljenja, vendar zajema več ravni abstrakcije.	Zgolj programiranje
Temeljna spretnost: gre za spretnost, ki jo v sodobni družbi za svoje delovanje potrebuje vsak posameznik.	Rutinsko mehansko delovanje
Način človekovega mišljenja	Način delovanja računalnika

Kaj je računalniško mišljenje?	Kaj ni računalniško mišljenje?
Ideje: produkcija idej, strategij in pristopov za reševanje problemov, za uravnavanje vsakodnevnega delovanja, komunikacije in medosebnih odnosov na način, da je rešitev algoritem, ki ga lahko izvedeta človek ali stroj.	Artefakti/izdelki: produkcija strojne in programske opreme
Pomembno za vsakogar in povsod: realnost, ki jo je treba vključiti v posameznikov način delovanja in pristopanja k reševanju problemov.	Pomembno zgolj za ožjo skupino (zainteresiranih) posameznikov, ki jih zanimata računalništvo in informatika.

Mark C. Baker je razvoj računalniškega mišljenja (Baker, 2019) razdelil na štiri področja: razvoj konceptov, spretnosti, značilnosti in pristopov.



Slika 1: Računalniško mišljenje (Baker, 2019)

2.1.2 Koncepti in pristopi računalniškega mišljenja

Med koncepte računalniškega mišljenja prištevamo naslednje:

- logično sklepanje: napovedovanje in analiziranje,
- postopkovno razmišljanje: zaporedje korakov in pravila,
- dekompozicija: razstavljanje problema na manjše probleme,
- abstrakcija: zanemarjanje podrobnosti in detajlov,
- iskanje vzorcev: podobnosti in posploševanje,
- vrednotenje rešitve.

Računalniško mišljenje je sestavljeno iz petih pristopov:

- preizkušanje,
- ustvarjanje,
- razhroščevanje,
- vztrajanje,
- sodelovanje.

Slika 2: Računalniški mislec: Koncepti in pristopi (Računalniški mislec, 2020)

2.2 Razvijanje računalniškega mišljenja

Curzon, Dorling, Ng, Selby in Woollard so pripravili okvir razvoja računalniškega mišljenja s pomočjo igre Minecraft Education Edition (Curzon, Dorling, Ng, Selby in Woollard, 2014).

Na podlagi pregleda akademskih referenc sta Selby in Woollard kot ključne predlagala naslednje koncepte:

- algoritmično razmišljanje,
- vrednotenje,
- dekompozicija,
- abstrakcija,
- posploševanje.

Tabela 2: Opis konceptov (Curzon idr., 2014)

Koncept	**Opis**
Algoritmično razmišljanje	Algoritmično razmišljanje je način, kako priti do rešitve z jasno opredelitvijo korakov. Namesto enega samega odgovora, npr. 42, s pomočjo algoritmičnega razmišljanja razvijemo sklop navodil ali pravil, s pomočjo katerih pridemo (bodisi človek, bodisi računalnik) do odgovora, če natančno upoštevamo omenjena navodila.
Vrednotenje	Vrednotenje je postopek, s katerim zagotovimo, da je algoritmična rešitev ustrezna. Različne lastnosti algoritmov je treba oceniti: ali so pravilni, dovolj hitri, ekonomični pri uporabi virov, so ljudem lahki za uporabo ipd. Treba je sprejemati kompromise, saj je le redko mogoče najti eno idealno rešitev za vse situacije.

Koncept	Opis
Dekompozicija	Dekompozicija je način razmišljanja o problemih in algoritmih, ki jih lahko razbijemo na manjše dele. Nato je mogoče ločene dele razumeti, rešiti, razviti in ovrednotiti ločeno. To olajša reševanje zapletenih problemov in načrtovanje obsežnih algoritmov.
Abstrakcija	Abstrakcija je še en način za lažje razmišljanje o problemih, ki vključuje skrivanje podrobnosti (delanje oboka v kockastem svetu). Spretnost je v izbiri pravih podrobnosti, ki jih želite skriti, tako da problem postane enostavnejši, ne da bi izgubili kaj pomembnega. Uporablja se kot način za lažje ustvarjanje zahtevnejših algoritmov.
Posploševanje	Posploševanje je način hitrega reševanja novih problemov na podlagi prejšnjih, že rešenih problemov. Vzamemo lahko algoritem, ki reši določen problem, in ga prilagodimo tako, da reši celo vrsto podobnih težav.

Curzon idr. navajajo, da so opisi zgornjih konceptov na visoki ravni. Čeprav so pomembni, sami ne razložijo, kako lahko razvoj računalniškega mišljenja vključimo v poučevanje in raziskovanje. Zato je pomembno še dodatno določiti tehnike za razvoj posameznih konceptov.

Tabela 3: Primeri tehnik za razvijanje konceptov računalniškega mišljenja (Curzon idr., 2014)

Koncept	Nekaj primerov tehnik
Algoritmično razmišljanje	Pisanje ukazov, ki dosežejo zaželeni učinek, če si sledijo v določenem vrstnem redu. Pisanje ukazov, ki uporabljajo aritmetične in logične operacije za dosego zaželenega učinka. Pisanje ukazov, ki shranjujejo, premikajo in upravljajo podatke, da dosežejo želeni učinek (spremenljivke). Združevanje ukazov v funkcije in uporaba funkcij. Pisanje sklopov navodil, ki jih lahko hkrati opravljajo različni agenti (računalniki ali ljudje) za dosego zaželenega učinka (vzporedno razmišljanje in obdelava, sočasnost). Ustvarjanje algoritmov za preizkušanje hipoteze.
Vrednotenje	Ocenjevanje, ali je algoritem primeren za svoj namen. Ocena, ali algoritem deluje pravilno (funkcionalna pravilnost). Oblikovanje in izvajanje načrtov, preizkusov in analiziranje rezultatov (testiranje). Primerjava zmogljivosti algoritmov, ki opravljajo enako nalogo. Sklepanje kompromisov med nasprotujočimi si zahtevami. Presoditi, kdaj je algoritmična rešitev dovolj dobra, četudi ni popolna. Ocenjevanje, ali rešitev ustreza specifikaciji (merilom).
Dekompozicija	Razčlenitev problemov na sestavne dele za lažje reševanje osnovnega problema. Razčlenitev problema na enostavnejše probleme, ki jih je mogoče rešiti na enak način (rekurzija).

Koncept	Nekaj primerov tehnik
Abstrakcija	Zmanjšanje zapletenosti z odstranjevanjem nepotrebnih podrobnosti. Izbira načina za predstavitev predmetov in procesov, ki omogočajo njihovo upravljanje na koristne načine. Skrivanje celotne kompleksnosti predmeta ali procesa. Skrivanje zapletenosti podatkov, na primer z uporabo podatkovnih struktur. Prepoznavanje razmerij med abstrakcijami. Filtriranje informacij pri razvoju rešitev.
Posploševanje	Ugotavljanje vzorcev in skupnih značilnosti problemov, procesov, rešitev ali podatkov. Prilagajanje rešitev ali delov rešitev, tako da veljajo za celo vrsto podobnih problemov. Prenos idej in rešitev iz enega problemskega področja v drugega.

2.3 Analiza prisotnosti razvijanja računalniškega mišljenja pri obveznih predmetih v OŠ

Najprej sem pregledal učne načrte za osnovno šolo v digitalni obliki (Interaktivni učni načrti, 2020) in preveril, pri katerih predmetih smo morebiti razvijali kakšen element računalniškega mišljenja. Ugotovil sem, da razvoj računalniškega mišljenja pri nas ni celovito vključen v učne načrte v osnovni šoli. Posamezni koncepti in elementi računalniškega mišljenja se sicer pojavljajo tudi pri obveznih predmetih, vendar ne v polnem obsegu in na način, da bi računalniško mišljenje razvijal sistematično in načrtno.

Tabela 4: Primeri dejavnosti v OŠ, pri katerih se pojavlja računalniško mišljenje

Koncept	Predmet	Primer
Algoritmično razmišljanje	MAT, FIZ, KEM, TIT	Pri računanju s pomočjo žepnega računala je pomembno ustrezno zaporedje uporabljenih tipk. Vrstni red korakov je pomemben v postopku obdelave materiala. Kemijski eksperimenti zahtevajo natančno sledenje načrtovanim korakom.
Vrednotenje	MAT	Za reševanje zahtevnejšega problema je običajno treba izvesti vrsto korakov (algoritem) v pravilnem vrstnem redu in ob koncu reševanja ovrednotiti ustreznost poti reševanja.
Dekompozicija	MAT	Pri reševanju zahtevnejših matematičnih problemov je pogosto treba problem razdeliti na manjše dele.
Abstrakcija	LUM	V abstraktnem slikarstvu lahko umetnik poenostavi izbrano podobo.
Posploševanje	MAT	Pri raziskovanju vzorcev lahko prepoznamo pravilo v vzorcu in poiščemo posplošitev (npr. z zapisom algebrskega izraza).

3 RAZISKOVALNI DEL

3.1 Metode dela

3.1.1 Merjenje

Po ugotovitvi, da ne obstajajo oziroma so načrti šolske zgradbe (predvsem starega dela) težje dostopni, je bila edina možnost šolo premeriti. Za merjenje sem uporabil laserski merilnik in meter.

3.1.2 Načrtovanje

Za lažje modeliranje šolske zgradbe v virtualnem svetu so osnova natančni načrti posameznih etaž. Zato sem pred programiranjem moral izrisati še načrte posameznih etaž v starem in novem delu šolske zgradbe s pomočjo brezplačnega programa Sweet Home 3D.

3.1.3 Modeliranje zgradbe iz realnega sveta v virtualnem svetu s programiranjem

Za gradnjo modela šole sem uporabil različico programa Minecraft Education Edition. Model šole sem gradil postopno po posameznih fazah. Gradnjo modela v virtualnem svetu sem opravil s pomočjo programiranja: najprej s pomočjo programiranja z bloki, kasneje pa s pomočjo programiranja v JavaScriptu. Sproti sem iskal tudi rešitve za izzive, ki so se pojavljali pri modeliranju. Razvoj računalniškega mišljenja je mogoč na vsakem koraku, ne samo pri programiranju, ampak tudi pri reševanju problemov, ki se pojavljajo sproti.

3.2 Uporabljena tehnična in programska oprema

3.2.1 Laserski merilnik razdalj

Laserski merilnik razdalj uporablja za merjenje razdalje svetlobni (laserki) žarek. S pritiskom na gumb merilnik pošlje laserski žarek proti merjeni točki. V merjeni točki se laserski žarek odbije in to odbito svetlobo sprejme fotocelica v merilniku. Merilnik nato izračuna razdaljo glede na časovni interval med pošiljanjem in sprejemom laserskega žarka v napravi. Največji prednosti merjenja s pomočjo takega merilnika sta njegova natančnost in prihranek časa.

Slika 3: Laserski merilnik razdalje

3.2.2 Meter

V določenih okoliščinah laserski merilnik ni bil ustrezen za merjenje, zato sem v takih primerih uporabil navadno merilo oziroma meter.

3.2.3 Aplikacija Sweet Home 3D

Sweet Home 3D je brezplačna aplikacija za notranje oblikovanje, s katero lahko rišemo načrte stavb, razporejamo stavbno in notranje pohištvo ter si ogledamo rezultate v 3D. Načrte je možno tudi kotirati in natisniti.

3.2.4 Minecraft Education Edition

Minecraft Education Edition je izobraževalna različica osnovne igre, zasnovana posebej za uporabo v izobraževalnih ustanovah in zgrajena iz kodne baze Bedrock. Na voljo je v operacijskih sistemih Windows 10, MacOS, iPadOS in Chrome OS. Vključuje paket virov za kemijo, brezplačne načrte lekcij na spletnem mestu Minecraft Education Edition in dve brezplačni spremljevalni aplikaciji: Code Connection in Classroom Mode.

Prva beta različica je bila objavljena sredi leta 2016. Polna različica igre je bila nato izdana v sistemih Windows 10 in MacOS novembra 2016 (Minecraft Education Edition, 2021).

Pomemben del Minecraft Education Edition je tudi razširitev Code Builder, ki omogoča enostavno vizualno programiranje z bloki ali programiranje v tekstovni obliki (JavaScript ali Python).

3.2.5 3D-slikar

Za izgled določenih gradbenih elementov v Minecraftu sem moral spremeniti izgled slik v t. i. Texture Packu. Texture Pack vsebuje dvodimenzionalne slike gradbenih elementov v obliki datotek PNG in lastnosti teh elementov (kot so velikost, oblika, orientacija in druge lastnosti) v obliki datotek JSON. Ker ni mogoče uporabiti lastnega Texture Packa, sem moral prilagoditi videz nekaterih gradbenih elementov v obstoječem Texture Packu. Spremembo videza sem izvedel s pomočjo aplikacije 3D-slikar.

3.2.6 Google Maps, Google Street View in Google Earth

Nekateri deli šolske zgradbe so zame fizično nedostopni, zato sem bil primoran za spoznavanje z izgledom šole uporabiti tudi Googlove aplikacije, ki omogočajo ogled stavb iz zraka. Prav tako mi je zelo pripomogel videoposnetek Velike Nedelje iz ptičje perspektive (Velika Nedelja iz ptičje perspektive, 2021).

4 REZULTATI IN RAZPRAVA

4.1 Koncepti računalniškega mišljenja skozi postopek gradnje šole v virtualnem okolju

4.1.1 Algoritmi kot element računalniškega mišljenja

Pri načrtovanju in pisanju programov sem moral najprej sestaviti algoritme, ki sem jih potem še sprogramiral v urejevalniku kode v Minecraftu.

Pri pripravi algoritmov sem uporabljal različne koncepte, kot so: zaporednost izvajanja ukazov, izvedba ponavljajočih operacij z zankami, ukazi z aritmetičnimi in logičnimi operacijami, uporabo spremenljivk ter uporabo podprogramov (funkcij).

4.1.2 Vrednotenje

Pri izbiri merila oz. razmerja med realno in virtualno velikostjo elementov sem spreminjal in prilagajal merilo, dokler nisem našel najustreznejšega. Pri tem sem uporabil koncept računalniškega mišljenja, in sicer vrednotenje.

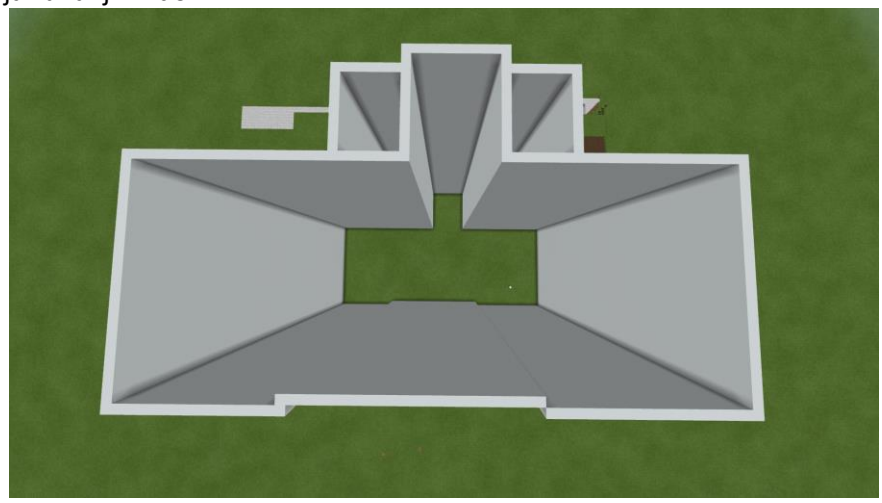
Prav tako sem po vrednotenju časa izgradnje ugotovil, da je agent neustrezen za gradnjo tako velikega virtualnega modela. Na osnovi tega sem poiskal drugo rešitev in namesto agenta uporabil "Builderja" oz. robota, ki izvaja ukaze nekajkrat hitreje kot agent.

Gradnjo šole sem nato razdelil na veliko manjših faz. Po pripravi dela programa za posamezno fazo sem izvedel gradnjo te faze, nato pa vizualno preveril ustreznost gradnje (in s tem ustreznost zapisanega programa). V kolikor sem ugotovil pomanjkljivosti, sem ustrezno popravil kodo in postopek po potrebi ponovil. Prav tako se pri vrsti drugih izzivov, ki so nastali v času pisanja kode, pojavi potreba po iskanju rešitve, vrednotenju rešitve in sklepanju kompromisov za izbiro najustreznejše rešitve.

4.1.3 Dekompozicija kot element računalniškega mišljenja

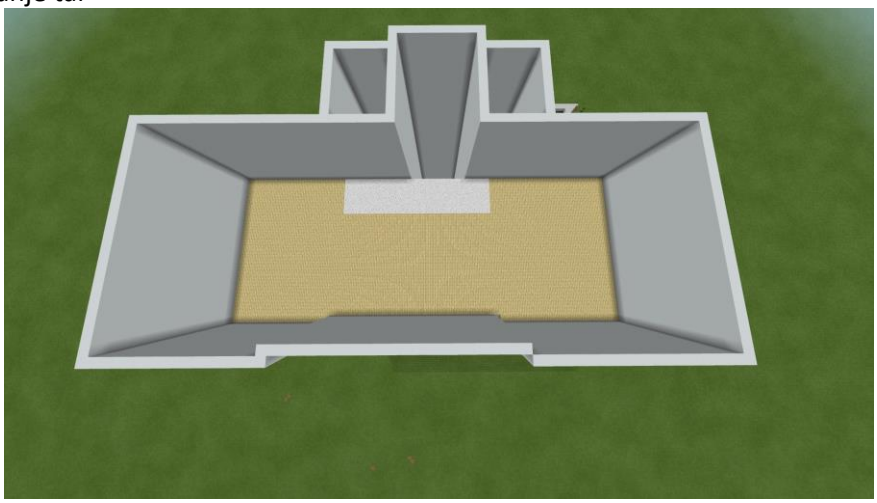
Velikonedeljska šola je objekt, ki je bil zgrajen v dveh fazah. V prvi fazi je bil zgrajen stari del šole, ki je sestavljen iz kleti, pritličja, prvega in drugega nadstropja ter podstrešne mansarde. V drugi fazi je bil zgrajen novejši del šole, ki obsega klet, prvo in drugo nadstropje ter knjižnico, pisarne, zbornico, atrij, jedilnico in kuhinjo. Že postopek gradnje šole v dveh fazah nakazuje na to, da je bilo treba tudi gradnjo v virtualnem svetu razdeliti na dva obsežnejša dela. Gradnjo posameznega dela šole pa je moč še dalje razdeliti na manjše procese, ki so:

a) gradnja zunanjih zidov



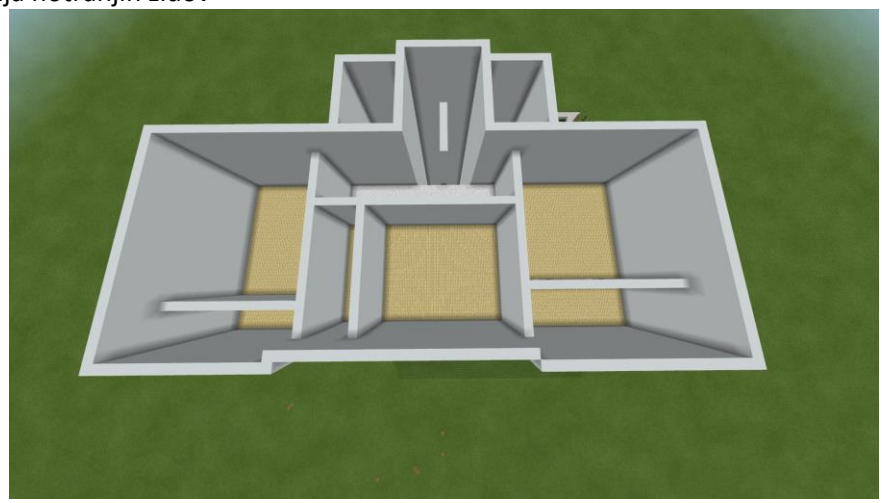
Slika 4: Pogled na zunanje zidove iz ptičje perspektive (stari del šole)

b) polaganje tal



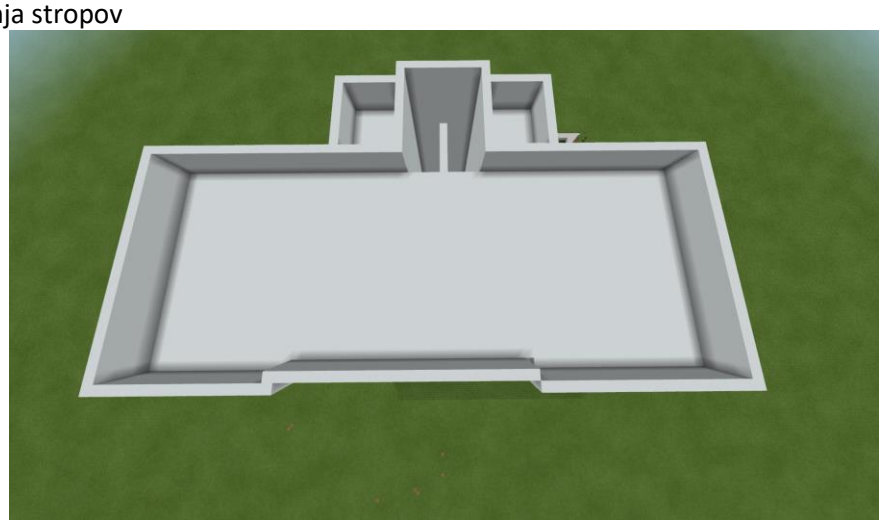
Slika 5: Pogled na tla v 2. nadstropju iz ptičje perspektive (stari del šole)

c) gradnja notranjih zidov



Slika 6: Pogled na notranje zidove 2. nadstropja iz ptičje perspektive (stari del šole)

d) gradnja stropov



Slika 7: Pogled na strop 2. nadstropja iz ptičje perspektive (stari del šole)

e) postavljanje oken

Slika 8: Pogled na okna (stari del šole)

f) postavljanje vrat

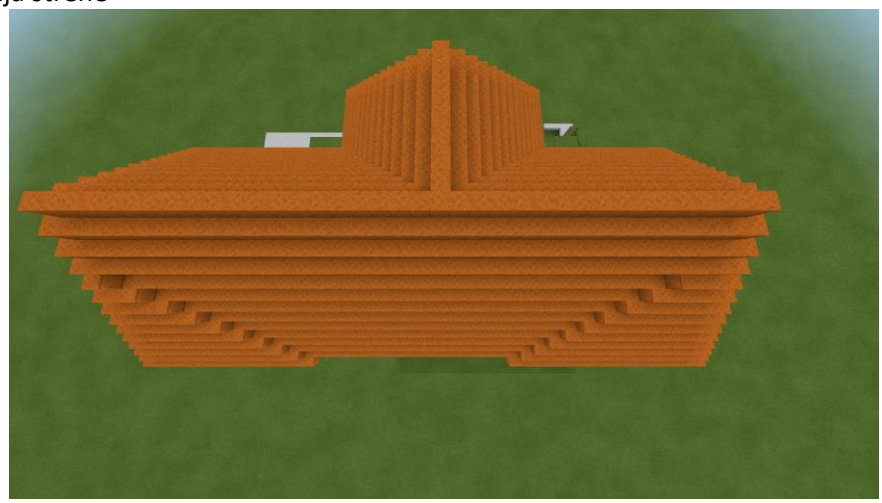
Slika 9: Pogled na hodnik in vrata (novi del šole)

g) gradnja stopnišč in podestov

Slika 10: Pogled na stopnišče in hodnik v (stari del šole)

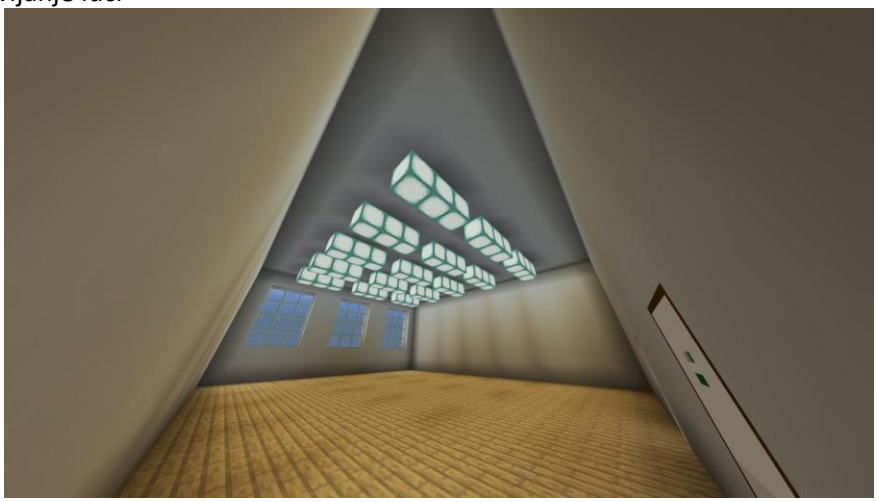
Slika 11: Pogled na podest stopnišča (stari del šole)

h) gradnja strehe



Slika 12: Pogled na streho iz ptičje perspektive (stari del šole)

i) postavljanje luči



Slika 13: Pogled na luči v učilnici (stari del šole)

Proti koncu sestavljanja kode sem ugotovil tudi, da je maksimalna dolžina kode programa v različici Minecraft EDU omejena. Zato sem moral celotno kodo za gradnjo virtualnega modela šole razdeliti na

tri manjše dele (samostojne programe) in nato v pravem zaporedju izvesti posamezen program iz omenjene trojice. Tudi pri tem postopku sem uporabil dekompozicijo.

4.1.4 Abstrakcija kot element računalniškega mišljenja

Zaradi zakonitosti v Minecraft-vem virtualnem svetu je bilo treba opustiti kar nekaj začetnih idej o usklajevanju podrobnosti z realnim svetom. V začetku sem želel uporabiti merilo, pri katerem ena kocka v virtualnem svetu predstavlja 25 cm v realnem svetu. Kmalu sem ugotovil, da takšno merilo povzroča veliko težav v nadaljnji gradnji, kot so npr. izjemno visoki prostori, ki jih ni mogoče ustrezno osvetliti, neustrezna velikost vrat ipd.

Zato sem merilo prilagodil tako, da ena kocka v virtualnem svetu predstavlja 50 cm v realnem svetu. Tudi takšno merilo ima seveda svoje pomanjkljivosti. Zato je bilo potrebnih nemalo prilagoditev.

Naj jih omenim samo nekaj:

- prilagoditev širine zidov (npr. s 60 cm na 50 cm),
- prilagoditev višine posamezne stopnice in števila stopnic,
- višina vrat je v virtualnem svetu natančno določena in sicer 2 kocki, podobno kot za višino velja za širino vrat (1 kocka),
- zaokrožitev širine oken (npr. z 80 cm na 100 cm).

V Minecraft Education Edition smo dodatno omejeni tudi glede nekaterih drugih detajlov, kot so npr. barve in materiali.

V koncept abstrakcije sodi tudi »skrivanje« kompleksnosti programiranja v podprograme (funkcije) in večkratno klicanje teh podprogramov. Namreč, ko sem enkrat pripravil (sestavil) ustrezen podprogram, se mi pri klicu tega podprograma ni bilo več potrebno ukvarjati s tem, kako podprogram deluje, ampak sem moral pri klicu podprograma le uporabiti ustrezne parametre. Primeri podprogramov oz. funkcij se nahajajo v naslednjem podrazdelku.

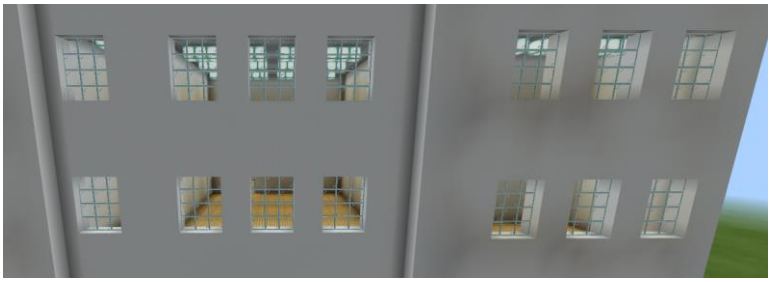
4.1.5 Posploševanje kot koncept znotraj računalniškega mišljenja

Pri gradnji se pojavlja veliko ponavljajočih se vzorcev v različnih fazah (tako v starem kot novem delu šole), npr.:

- Notranje stene v prvem in drugem nadstropju starega dela so enako razporejene.
- Stopnišča v starem delu so enaka v vseh nadstropjih razen v kleti.
- Na strehi v starem in novem delu se pojavljajo deli strehe v obliki paralelogramov in trikotnikov.
- Učilnice v novem delu so povsem enako razporejene v vseh nadstropjih.
- Luči v prvem in drugem nadstropju starega dela so v prostorih enako razporejene, podobno velja tudi za luči v vseh nadstropjih novega dela šole.

Posebej bi izpostavil naslednje vzorce:

a) vzorec oken v 1. in 2. nadstropju starega dela šole

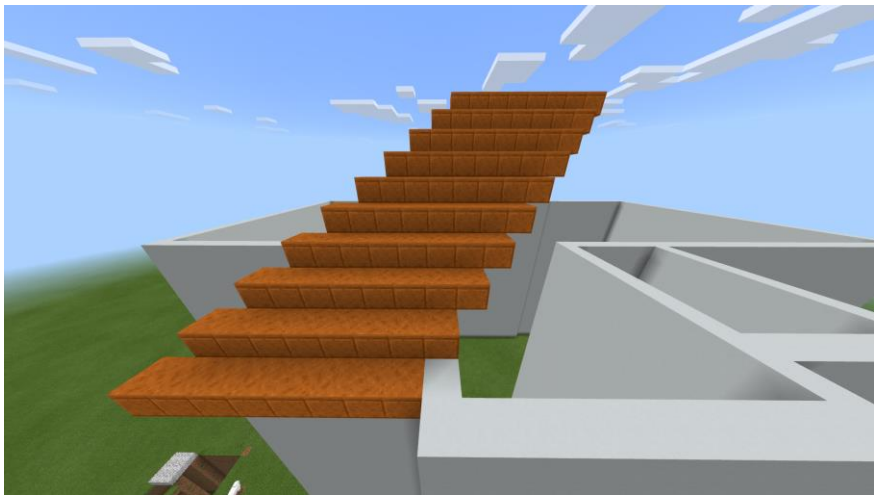


Slika 14: Vzorec oken v 1. in 2. nadstropju (stari del šole)

b) vzorec oken v vseh nadstropjih novega dela šole

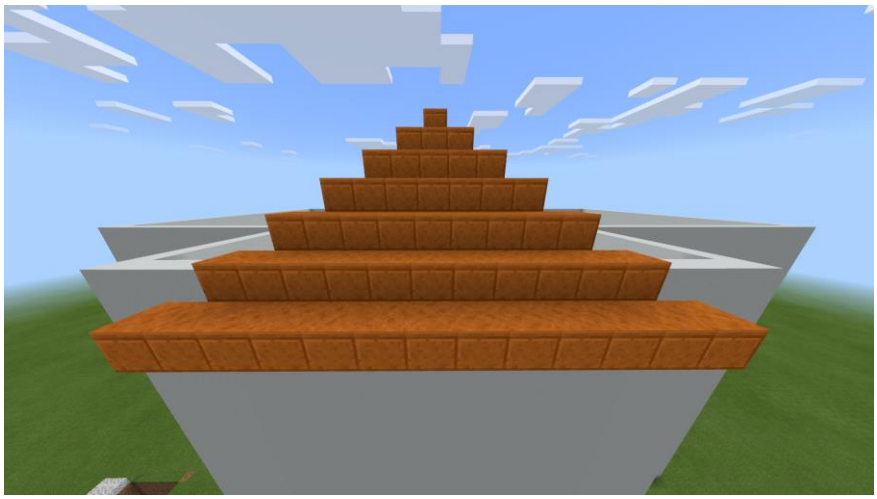
Slika 15: Vzorec oken (novi del šole)

c) del strehe v obliki paralelograma



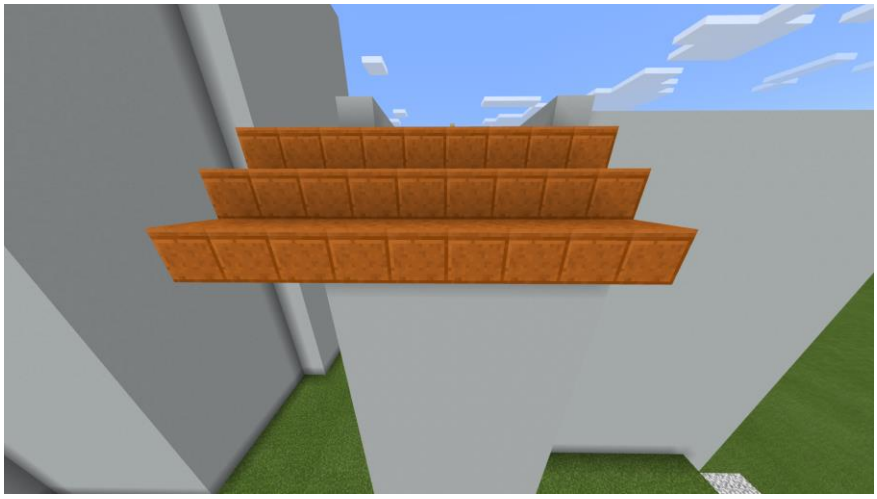
Slika 16: Pogled na del strehe v obliki paralelograma (stari del šole)

d) del strehe v obliki trikotnika



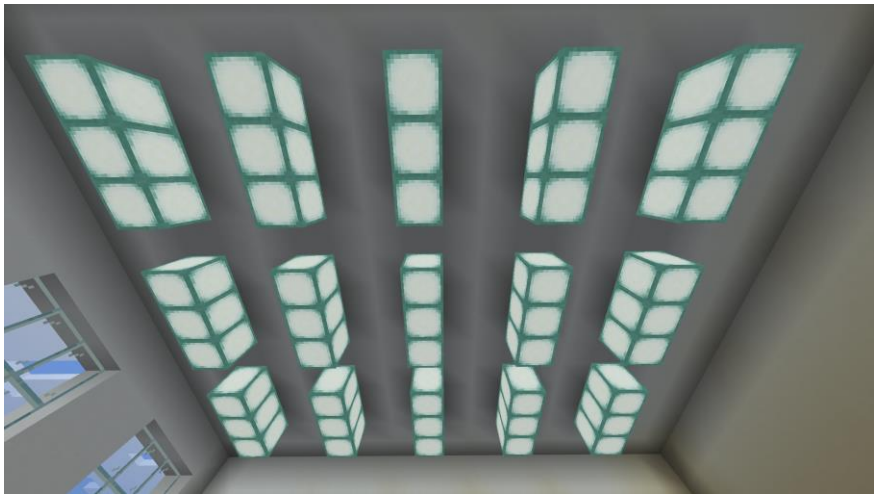
Slika 17: Pogled na del strehe v obliki trikotnika (stari del šole)

e) del strehe v obliki pravokotnika



Slika 18: Pogled na del strehe v obliki pravokotnika (novi del šole)

f) vzorec luči v učilnici



Slika 19: Vzorec luči v eni izmed učilnic (stari del šole)

Podprogrami za gradnjo posameznih vzorcev se nahajajo v prilogi.

4.2 Posebni izzivi pri gradnji virtualnega modela šole

4.2.1 Izbira merila

Prvotno sem izbral merilo, pri katerem štiri kocke v virtualnem svetu predstavljajo en meter v naravi, vendar sem opazil, da bodo prostori zelo visoki, kar bi povzročilo težave z osvetlitvijo. Zaradi tega sem se odločil za merilo, pri katerem dve kocki v virtualnem svetu predstavljata en meter v naravi.

4.2.2 Prilagoditev gradbenih elementov in uporabljeni gradbeni elementi

Prvotna ideja je bila sestaviti lasten t. i. Texture Pack, ki bi vseboval gradbene elemente, podobne tistim v realnem svetu. Pri uvozu lastnih Texture Packov pa se v različici Minecraft EDU pojavlja težava, ki je kljub raziskovanju zapisov na internetu in postavljanju vprašanj na forumih nisem uspel rešiti. Po uvozu lastnega Texture Packa se namreč gradbeni elementi iz tega paketa ne pojavijo v inventarju. Zaradi

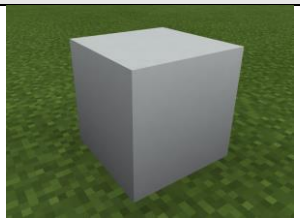
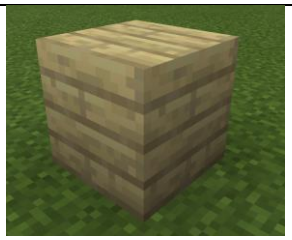
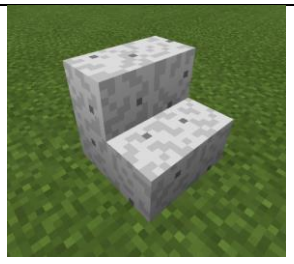
navedenega sem opustil prvotno idejo in zgolj prilagodil nekatere gradbene elemente v obstoječem privzetem Texture Packu. Spremembe sem opravil s pomočjo aplikacije 3D-slikar. Takšen primer so notranja lesena vrata (vstopna vrata v učilnice).

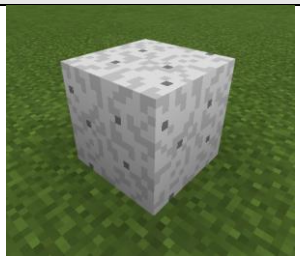
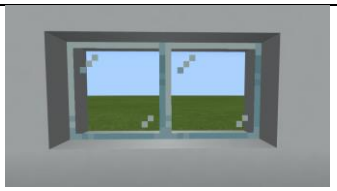
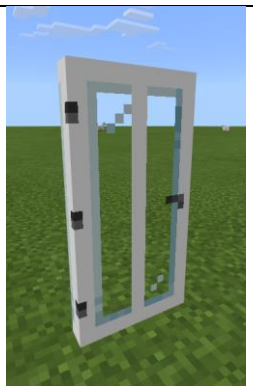
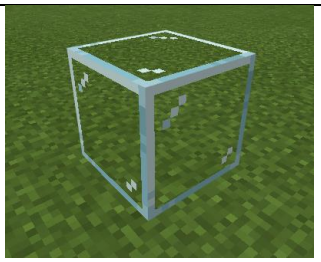
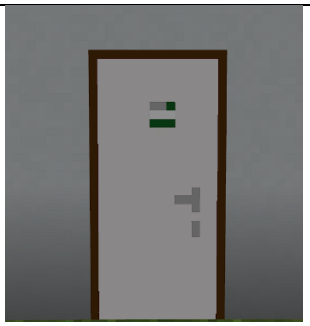
Slika 20: Originalni izgled vrat v Minecraftu (levo) in spremenjen izgled (desno)

Spremembo videza sem opravil tudi na nekaterih drugih gradbenih elementih. Ni pa mi uspelo prilagoditi osnovnega gradbenega elementa - tj. zidaka za stene. V privzetem Texture Packu ni kocke, ki bi jo lahko pobarval na zunanji strani z eno in na notranji strani z drugo barvo. Tak zidak bi potreboval za gradnjo zunanjih zidov (rumena fasada in bela barva v notranjosti). Iz omenjenega razloga je tudi zunanjost virtualnega modela šole bele barve.

Spodnja preglednica prikazuje vse gradbene elemente, ki sem jih uporabil za gradnjo šole, njihov naziv (ime) v Minecraftu in njihov izgled.

Tabela 5: Uporabljeni gradbeni elementi in njihov izgled

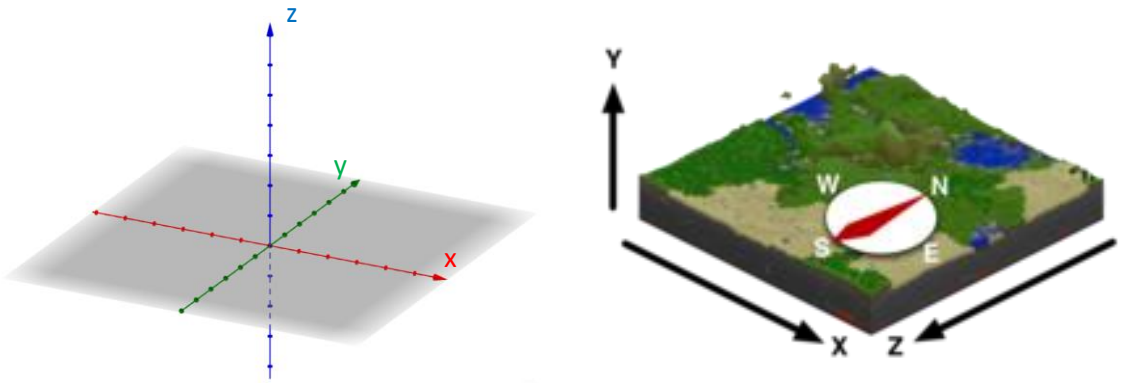
Gradbeni element	Naziv v Minecraftu	Izgled
Zidak za stene	White concrete	
Parket	Birch wood planks	
Stopnice	Sandstone stairs (sprememba izgleda s 3D-slikarjem)	

Gradbeni element	Naziv v Minecraftu	Izgled
Talna obloga na hodnikih	Cut sandstone (sprememba izgleda s 3D-slikarjem)	
Okna	Glass pane	
Steklena vrata	Acacia door (sprememba izgleda s 3D-slikarjem)	
Obroba steklenih vrat	Glass	
Vrata notranja, lesena	Oak door (sprememba izgleda s 3D-slikarjem)	

Gradbeni element	Naziv v Minecraftu	Izgled
Vrata zunanja, lesena	Dark oak door	

4.2.3 Koordinatni sistem

Koordinatni sistem v Minecraftu se nekoliko razlikuje od nam znanega koordinatnega sistema v prostoru. Koordinate temeljijo na mreži, kjer se na izhodišču sekajo tri črte ali osi. Os x prikazuje igralčevo razdaljo vzhodno (pozitivno) ali zahodno (negativno) od izhodiščne točke - tj. zemljepisno dolžino. Os z označuje igralčevo oddaljenost južno (pozitivno) ali severno (negativno) od izhodiščne točke - tj. zemljepisno širino. Koordinata y pove, kako visoko od 0 do 255 je trenutno igralec oz. pomeni, kako visoko bo postavljen gradbeni element (Coordinates, 2021).



Slika 21: Matematični koordinatni sistem v prostoru (levo) in koordinatni sistem v Minecraftu (desno)

Pri primerjavi koordinatnega sistema v prostoru, ki ga uporabljam pri matematiki, in tega, ki ga uporablja Minecraft sem opazil, da sta koordinati y in z zamenjani. Kljub temu sem pri sestavljanju podprogramov (funkcij) uporabljal matematični koordinatni sistem, npr. v podprogramu za postavljanje oken `function okna (x0: number, y0: number, z0: number, x1: number, y1: number, z1: number)`. Minecraftov koordinatni sistem pa sem bil primoran uporabiti pri klicanju neposrednih ukazov, kot npr. ukaza `blocks.fill`.

4.2.4 Časovna potratnost gradnje s pomočjo agenta

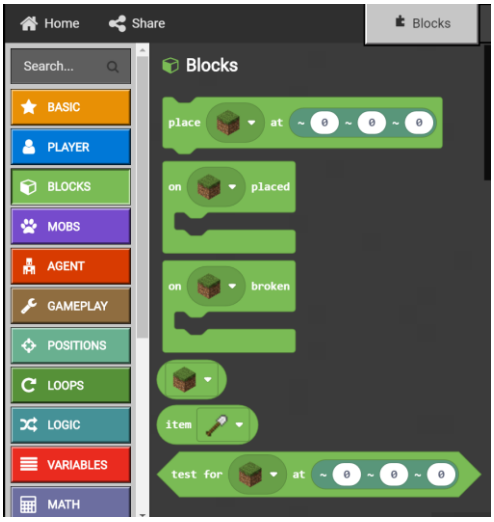
V različici igre Minecraft Education Edition lahko gradbene elemente postavljamo na tri načine. Prvi način je ta, da gradnjo izvede agent po sestavljenih ukazih.

Agent oz. pomočnik spada med t. i. mob-e, ki so v Minecraftu živa bitja, ki se v igri premikajo. Izraz "mob" je okrajšava za izraz mobilen in se v osnovni različici igre pogosto uporablja za živa, gibljiva bitja, kot so npr. netopirji, piščanci, krave, konji, gobe, prašiči, zajci, ovce, snežaki, lignji, vaščani, plameni,

plazilci, zmaji, duhovi, velikani, okostnjaki, sluzi, čarovnice, zombiji, zombirani pujsi, pajki in volkovi (Mobs in Minecraft, 2021).

Slika 22: Izgled agenta (levo) in nekateri ukazi, ki jih lahko agent izvede (desno)

Drugi način postavljanja gradbenih elementov je uporaba ukazov iz sklopa Blocks v Code Builderju. Ob uporabi teh ukazov gradbenih elementov ne postavlja niti agent niti igralec, ampak neke vrste robot, ki je del Code Builderja (v nadaljevanju robot).



Slika 23: Nekateri ukazi za delo z gradbenimi elementi

Gradnjo šole sem najprej začel programirati v prvem načinu. Vendar sem ugotovil, da agent gradi prepočasi in bi izgradnja celotne šole na tak način trajala zelo dolgo. Dodatna težava je tudi ta, da je treba pri tako obsežni gradnji posamezne dele programa zaganjati velikokrat, jih prilagajati, popravljati in posodabljati, kar bi ponovno zahtevalo veliko časa (tudi čakanja).

Kljub prej omenjenima načinoma gradnje pa je treba pri postavljanju posameznih gradbenih elementov uporabiti še tretji način. To pa je gradnja s pomočjo ukazov iz t. i. konzole. Namen teh ukazov je razložen v nadaljevanju.

4.2.5 Postavljanje gradbenih elementov s pomočjo ukazov iz konzole

Gradbene elemente robot vedno postavlja od zahoda proti vzhodu. Zaradi tega nastanejo posamezne težave pri gradnji elementov, ki morajo biti obrnjeni v kateri izmed drugih smeri, npr. stopnišča. Po iskanju informacij na internetu sem ugotovil, da je mogoče elemente postavljati v različnih smereh s pomočjo ukazov v konzoli. Spodnja preglednica prikazuje nabor konzolnih ukazov za gradnjo stopnic v različnih smereh.

Tabela 6: Primeri konzolnih ukazov za postavljanje gradbenih elementov v različnih smereh

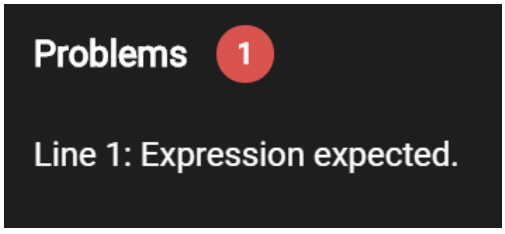
Ukaz	**Smer postavitve gradbenega elementa**
`/fill 0 4 0 0 4 0 birch_stairs 0`	Od zahoda proti vzhodu
`/fill 0 4 0 0 4 0 birch_stairs 1`	Od vzhoda proti zahodu
`/fill 0 4 0 0 4 0 birch_stairs 2`	Od severa proti jugu
`/fill 0 4 0 0 4 0 birch_stairs 3`	Od juga proti severu

Konzolni ukazi se pričnejo s poševnico (angl. slash), ki ji sledi ime ukaza, za tem pa še parametri. V ukazu "fill" so prvi trije parametri koordinate začetnega mesta postavljanja gradbenega elementa, drugi trije pa končnega mesta postavljanja. Ukaz fill lahko namreč postavi celo vrsto gradbenih elementov od začetnih do končnih koordinat. Za koordinatami kot parametra sledita še vrsta gradbenega elementa in številka, ki določa smer postavitve gradbenega elementa.

Podobna težava kot pri postavljanju stopnic se pojavlja tudi pri postavljanju vrat.

4.2.6 Blokovno ali tekstovno programiranje

Pri začetnih korakih gradnje šole v virtualnem svetu sem uporabil blokovno programiranje, ki sem ga poznal že od prej iz drugih blokovnih programskih jezikov (Scratch). Sčasoma sem ugotovil, da je struktura blokov pri gradnji tako velikega objekta postala preobsežna in neobvladljiva, zato sem se bil primoran začeti učiti tekstovno programiranje v obliki JavaScripta. Dele programa, ki sem jih sestavil v obliki blokov, Minecraft Education Edition samodejno pretvori v obliko JavaScripta. Zato mi ni bilo potrebno vnaprejšnje učenje sintaktičnih pravil, ampak se jih je mogoče učiti kar sproti med programiranjem. Minecraft EDU Code Builder sproti opozarja na morebitne sintaktične napake v programu, zato jih je precej lažje odpravljati.



Slika 24: Opozorilo v Code Builderju o napaki v kodi

5 ZAKLJUČEK/SKLEPI

Po tem, ko sem se odločil za temo raziskovalne naloge, sem bil mnenja, da se na poti do cilja ne bo pojavljalo veliko zahtevnih izzivov. V času izdelave raziskovalne naloge pa sem ugotovil, da je teh izzivov veliko in da so vsi izmed njih na tak ali drugačen način premagljivi. Nekateri izzivi, ki so se pojavljali, so bili resnično zahtevni in so zahtevali ogromno časa in energije za premagovanje. Nekaj teh izzivov sem že omenil v tej raziskovalni nalogi, še posebej pa je treba omeniti nedostopnost načrtov, težave z merjenjem, težave z gradnjo stopnic v virtualnem svetu, manjkajoče znanje s področja računalništva ter zelo slabo oz. sploh nerazvito računalniško mišljenje…

Nekatere hipoteze, ki sem jih postavil pred pričetkom gradnje sem potrdil, nekatere pa ovrgel.

1. *Izgradnjo šolske stavbe v Minecraftu bom lahko izvedel na osnovi načrtov šolske zgradbe.*
   Natančnih načrtov (predvsem starega dela šole) ni bilo mogoče v celoti pridobiti, zato je bilo smiselno uporabiti metodo merjenja prostorov in načrtovanja s pomočjo aplikacije Sweet Home 3D. Torej je ta hipoteza ovržena.
2. *V času rednega pouka nisem pridobil dovolj znanj za gradnjo virtualnega modela šole.*
   Ta hipoteza je potrjena, saj se v času rednega pouka nisem naučil nekaterih stvari, ki so zelo pomembne za gradnjo virtualnega modela šole, npr. računalniško programiranje in risanje načrtov.
3. *V času rednega pouka sem pridobil določene digitalne kompetence pri uporabi računalnika ter v manjšem obsegu tudi koncepte računalniškega mišljenja pri posameznih rednih predmetih.*
   Skozi osnovnošolsko izobraževanje sem pridobil osnovne digitalne kompetence, kot so npr. iskanje in vrednotenje informacij na spletu, uporaba digitalne identitete, uporaba programske opreme ipd. Koncepte računalniškega mišljenja sem v manjšem obsegu spoznal pri pouku matematike, fizike in likovne umetnosti. Iz zapisanega lahko sklepam, da je omenjena hipoteza potrjena.
4. *Manjkajo mi predvsem znanja s področja računalniškega programiranja.*
   V času gradnje virtualnega modela sem moral spoznati in usvojiti veliko večino konceptov računalniškega mišljenja, brez katerih mi ne bi uspelo zgraditi modela šole v Minecraftu. Zato lahko to hipotezo potrdim.
5. *Gradnjo celotne šole bo mogoče sprogramirati zgolj s pomočjo blokov.*
   To hipotezo lahko ovržem, saj sem najprej domneval, da bo mogoče gradnjo cele šole sprogramirati v blokovnem jeziku. Kmalu po začetku sestavljanja programa pa sem ugotovil, da bo koda v obliki blokov težko obvladljiva, zato sem se odločil uporabiti tekstovno programiranje v JavaScriptu.
6. *Nekatera stopnišča v šoli so sestavljena iz enakega števila stopnic z enako višino, zato predstavljajo ponavljajoči se vzorec.*
   Stopnišča v šoli predstavljajo primer vzorca. Torej lahko hipotezo potrdim. Se pa vzorci pri gradnji šole pojavljajo tudi drugod in ne zgolj na stopniščih. Primeri vzorcev so npr. tudi okna v posameznih nadstropjih tako starega kot tudi novega dela, v nekaterih učilnicah so luči razporejene po enakem vzorcu, nekatere notranje stene se pojavljajo v enakih oblikah na enakih mestih, vrata na omenjenih notranjih stenah so prav tako na enakih mestih.
7. *Pri gradnji posameznega dela šole bo obstajala zgolj ena programska rešitev, če zanemarimo vrstni red korakov.*
   Tudi zadnjo hipotezo lahko ovržem, saj je mogoče posamezne elemente postavljati v virtualnem svetu Minecraft EDU s pomočjo konzolnih ukazov, ukazov za agenta in ukazov za robota. To pa pomeni, da lahko obstaja več programskih rešitev.

Zastavljeni cilj, to je izgradnja virtualnega modela šole, sem dosegel. Celotna koda vseh treh programov je na voljo na povezavi **https://tinyurl.com/4j8banet**. Slike virtualnega modela šole so v prilogi.

Na poti do cilja sem raziskal področje računalniškega mišljenja ter ugotovil, da vsebin, ki bi to področje razvijale v večjem obsegu, ne najdemo pri pouku rednih predmetov v osnovni šoli. Pokazal sem, da je programiranje v Minecraftu lahko odličen način za usvajanje konceptov in razvoj računalniškega mišljenja.

V raziskovalno nalogo je bilo vloženega veliko časa in truda, še vedno pa obstajajo možnosti nadaljnjega razvoja raziskave:

a) V smislu gradnje virtualnega modela bi lahko izvedel optimizacije kode (skrajšanje kode, zmanjšanje časovne zahtevnosti za izgradnjo modela).
b) Virtualni model bi lahko razširil še na urejanje notranjega pohištva, urejanje okolice in postavitev bližnjih okoliških zgradb (cerkev Svete trojice in križniškega gradu pri Veliki Nedelji).
c) Vključene koncepte računalniškega mišljenja bi lahko podrobneje predstavil skozi postopke gradnje virtualnega modela.
d) S pomočjo zgrajenega virtualnega modela šole bi lahko pripravil interaktivni vodič za razvijanje računalniškega mišljenja s programiranjem v Minecraftu.
e) Na osnovi zgrajenega modela bi lahko pripravil tudi virtualni ogled šole.

6 VIRI IN LITERATURA

Baker, M. C. (2019). The little book of computational thinking. Pridobljeno s http://www.educationvision.co.uk/evcarticle-computationalthinking6.pdf.

Coordinates. (b. d.) Na Wikipedia.com. Pridobljeno 20. marca 2021 s https://minecraft.fandom.com/wiki/Coordinates

Curzon, P., Dorling, M., Ng, T., Selby, C. in Woollard, J. (2014). Developing computational thinking in the classroom: a framework. Pridobljeno s https://eprints.soton.ac.uk/369594/1/DevelopingComputationalThinkingInTheClassroomaFramework.pdf.

Interaktivni učni načrti. (b. d.) Na Dun.zrss.augmentech.si. Pridobljeno 16. decembra 2020 s https://dun.zrss.augmentech.si/#/

ISTE in CSTA. (2011). Computational thinking in K–12 education leadership toolkit. Pridobljeno s https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf

Kranjc, R., Košir, K. in Čotar Konrad, S. (2017). Računalniško mišljenje – kaj je to in zakaj bi ga sploh potrebovali? *Vzgoja in izobraževanje, 48(4)*, 9-19.

Minecraft Education Edition. (b. d.) Na Wikipedia.com. Pridobljeno 6. marca 2021 s https://en.wikipedia.org/wiki/Minecraft#Education_Edition

Mobs in Minecraft. (b. d.) Na DigMinecraft.com. Pridobljeno 7. marca 2021 s https://www.digminecraft.com/mobs/index.php

Računalniški mislec [slika na spletu]. (2014). Pridobljeno 17. decembra 2020 s https://1.bp.blogspot.com/-P_E0RcsW9Fo/VqqYgdAO8TI/AAAAAAAAIFQ/1mxd0Z5wcoM/s640/ct.png

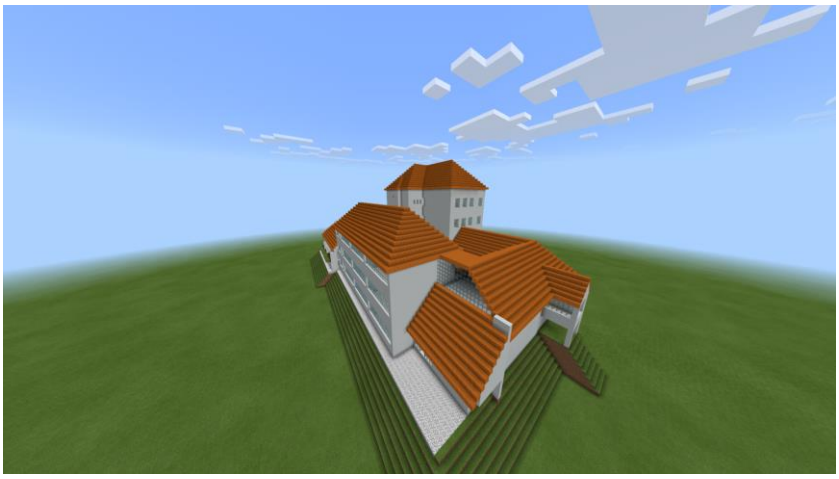
Rajšp, I. (19. julij 2020). *Velika Nedelja iz ptičje perspektive* [video datoteka]. Pridobljeno 20. januarja 2021 s https://www.youtube.com/watch?v=M_Oz5POn2so

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49(3)*, 33-35.

7 PRILOGE

PRILOGA A: Slike končanega virtualnega modela šole

Slika 25: Pogled na šolo (Velika Nedelja iz ptičje perspektive, 2021)



Slika 26: Pogled na zgrajen virtualni model šole (primerjava s sliko 25)

Slika 27: Pogled na zgrajen virtualni model šole (atrij)

Slika 28: Pogled na zgrajen virtualni model šole (od leve proti desni: kuhinja, jedilnica in novi del šole ter v ozadju stari del šole)

Slika 29: Pogled na zgrajen virtualni model šole (od leve proti desni: administrativni del, knjižnica in stari del šole)

Slika 30: Pogled na zgrajen virtualni model šole (od leve proti desni: stari del, kuhinja in jedilnica)

PRILOGA B: Podprogram za gradnjo vzorca oken v 1. in 2. nadstropju starega dela šole

```
function okna_n1_n2 () {
//okna v 1. in 2. nadstropju starega dela
    visina1 = nivo_z + 20, visina2 = nivo_z + 23
    for (let index = 0; index < 2; index++) {
        okna(22, 1, visina1, 20, 1, visina2)
        okna(17, 1, visina1, 15, 1, visina2)
        okna(12, 1, visina1, 10, 1, visina2)
        okna(2, 19, visina1, 2, 21, visina2)
        okna(2, 36, visina1, 2, 38, visina2)
        okna(2, 31, visina1, 2, 33, visina2)
        okna(2, 26, visina1, 2, 28, visina2)
        okna(1, 57, visina1, 1, 59, visina2)
        okna(1, 52, visina1, 1, 54, visina2)
        okna(1, 47, visina1, 1, 49, visina2)
        okna(25, 46, visina1, 25, 48, visina2)
        okna(25, 51, visina1, 25, 53, visina2)
        okna(25, 56, visina1, 25, 58, visina2)
        okna(25, 19, visina1, 25, 20, visina2)
        okna(22, 61, visina1, 20, 61, visina2)
        okna(17, 61, visina1, 15, 61, visina2)
    visina1 = nivo_z + 29 , visina2 = nivo_z + 32
    }
}
```

PRILOGA C: Podprogram za gradnjo vzorca oken v novem delu šole

```
function okna_nd () {
//okna v novem delu
    visina1 = nivo_z + 7, visina2 = nivo_z + 10
        for (let index1 = 0; index1 < 2; index1 ++) {
            okna(45, -7, visina1, 45, 0, visina2)
            okna(45, 9, visina1, 45, 16, visina2)
    visina1 = nivo_z + 15, visina2 = nivo_z + 18
    }
    visina1 = nivo_z + 15, visina2 = nivo_z + 18
        okna(45, 25, visina1, 45, 30, visina2)
    visina1 = nivo_z - 1, visina2 = nivo_z + 2
        for (let index2 = 0; index2 < 3; index2 ++) {
            okna(65, 34, visina1, 65, 32, visina2)
            okna(65, 30, visina1, 65, 15, visina2)
            okna(65, 13, visina1, 65, -2, visina2)
            okna(65, -4, visina1, 65, -19, visina2)
            okna(65, 36, visina1, 65, 42, visina2)
    visina1 = visina1 + 8, visina2 = visina2 + 8
    }
    visina1 = nivo_z + 2
        okna (45, -7, visina1, 45, 0, visina1)
        okna (45, 9, visina1, 45, 12, visina1)
        okna(46, 46, nivo_z + 12, 50, 46, nivo_z + 15)
}
```

PRILOGA D: Podprogram za postavljanje dela strehe v obliki paralelograma

```
function streha_paralel(x0: number, y0: number, z0: number, l: number,
sv: number, dir: number) {
//funkcija postavi del strehe v obliki paralelograma
    //x0, y0, z0 so začetne koordinate
    //l je dolžina vrste blokov
    //sv je število vrst
    //dir je smer postavljanja dela strehe
    //dir; 0-pomik v levo nagib zahod
    //    1-pomik v desno nagib zahod
    //    2-pomik v desno nagib vzhod
    //    3-pomik v levo nagib vzhod
    //    4-pomik v levo nagib sever
    //    5-pomik v desno nagib sever
    //    6-pomik v levo nagib jug
    //    7-pomik v desno nagib jug
    for (let index = 0; index < sv; index ++) {
        if (dir < 4) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0, z0, y0 + l)
            )
        }
        else if (dir > 3) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0 + l, z0, y0)
            )
        }
        if (dir == 0) {y0 ++, x0 --}
        else if (dir == 1) {y0 --, x0 --}
        else if (dir == 2) {y0 ++, x0 ++}
        else if (dir == 3) {y0 --, x0 ++}
        else if (dir == 4) {y0 --, x0 --}
        else if (dir == 5) {y0 ++, x0 --}
        else if (dir == 6) {y0 ++, x0 ++}
        else if (dir == 7) {y0 ++, x0 --}
        z0 ++
    }
}
```

PRILOGA E: Podprogram za postavljanje dela strehe v obliki trikotnika

```
function streha_triag(x0: number, y0: number, z0: number, l: number,
dir: number) {
//funkcija postavi del strehe v obliki trikotnika
    //x0, y0, z0 so začetne koordinate
    //l je dolžina prve vrste blokov
    //dir je smer postavljanja dela strehe
    //dir; 0-smer vrste sever jug in nagib proti zahodu
    //     1-smer vrste vzhod zahod in nagib proti jugu
    //     2-smer vrste sever jug in nagib proti vzhodu
    //     3-smer vrste vzhod zahod in nagib proti severu
    for (let index = l; index > - 1; index = index - 2) {
        if (dir == 0) {
        shapes.line(
        SMOOTH_RED_SANDSTONE,
        world(x0, z0, y0),
        world(x0, z0, y0 + index)
        )
        x0 --, y0 ++, z0 ++
        }
        else if (dir == 1) {
        shapes.line(
        SMOOTH_RED_SANDSTONE,
        world(x0, z0, y0),
        world(x0 - index, z0, y0)
        )
        x0 --, y0 ++, z0 ++
        }
        else if (dir == 2) {
        shapes.line(
        SMOOTH_RED_SANDSTONE,
        world(x0, z0, y0),
        world(x0, z0, y0 + index)
        )
        x0 ++, y0 ++, z0 ++
        }
        else if (dir == 3) {
        shapes.line(
        SMOOTH_RED_SANDSTONE,
        world(x0, z0, y0),
        world(x0 - index, z0, y0)
        )
        x0 --, y0 --, z0 ++
        }
    }
}
```

PRILOGA F: Podprogram za postavljanje dela strehe v obliki pravokotnika

```
function streha_pravo(x0: number, y0: number, z0: number, l: number,
sv: number, dir: number) {
//funkcija postavi del strehe v obliki pravokotnika
    //x0, y0, z0 so začetne koordinate
    //l je dolžina vrste blokov
    //sv je število vrst
    //dir je smer postavljanja dela strehe
    //dir; 0-nagib proti sever
    //     1-nagib proti jugu
    //     2-nagib proti vzhodu
    //     3-nagib proti zahod

    for (let index = 0; index < sv; index ++) {
            if (dir == 0) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0 + l, z0, y0)
            )
            y0 --, z0 ++
        }
        else if (dir == 1) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0 - l, z0, y0)
            )
            y0 ++, z0 ++
        }
        else if (dir == 2) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0, z0, y0 + l)
            )
            x0 ++, z0 ++
        }
        else if (dir == 3) {
            shapes.line(
            SMOOTH_RED_SANDSTONE,
            world(x0, z0, y0),
            world(x0, z0, y0 - l)
            )
            x0 --, z0 ++
        }
    }
}
```

PRILOGA G: Podprogram za postavljanje vzorca luči v učilnici

```
function luci_sj (x0: number, y0: number, z0: number, dcv: number,
scv: number, sv: number, pr_x: number, pr_y: number, pr_z: number,
nad: number) {
//funkcija postavi luči v smeri sever-jug
    //celota je skupek luči, ki se drži skupaj
    //x0, y0, z0-začetne koordinate
    //dcv-dolžina celote v vrsti
    //scv-število celot v vrsti
    //sv-število vrst
    //pr_x-preskok po x
    //pr_y-preskok po y
    //pr_z-preskok po z
    //nad-število nadstropij

    let x = x0
    let y = y0
    let z = z0
    for (let index3 = 0; index3 < nad; index3 ++) {
        for (let index2 = 0; index2 < sv; index2 ++) {
            for (let index1 = 0; index1 < scv; index1 ++) {
                blocks.fill(SEA_LANTERN, world(x, z, y), world(x, z, y + dc
v - 1))
                y = y + pr_y + dcv
            }
            y = y0
            x = x + pr_x
        }
        x = x0
        z = z + pr_z
    }
}
```

PRILOGA H: Primer tlorisa etaže v starem delu šole (pripravljenega s Sweet Home 3D)