Končni rezultati testiranj in uporabe distribuiranega sistema ART, navodila za uporabo

Tehnično poročilo A2.4

Revizija:	1.0
Projekt:	ARRS-ART
Avtorji:	Vito Čuček, Marjan Šterk, Gregor Pipan
Datum:	14. marec 2019

Kazalo

Kazalo slik

Kazalo primerov

Kazalo vstopnih točk aplikacijskega vmesnika

Slovar izrazov

Hypervisor	Program za upravljanje in izgradnjo virtualnih naprav.
GPU	Grafični procesor, sestavljen iz večjega števila manjših procesnih enot, specializiranih za aritmetične operacije.
CUDA	Programski jezik, prilagojen za programiranje grafičnih procesorjev NVIDIA.
Instanca	Manifestacija programske komponente ali virtualnega stroja na podlagi programske kode ali slike.
Node	Skupek virtualnih ali fizičnih naprav, za delovanje neodvisen od ostalih, v omrežju.
CPU	Centralna procesorska enota.
OptiX	Knjižnica CUDA za lažji razvoj algoritmov sledenja žarkom.
API	Aplikacijski vmesnik.
Komponenta	Logično ločena programska funkcionalnost večjega sistema.

1 Uvod

Tehnično poročilo je izdelano kot rezultat programskega sklopa 2.4 manjšega aplikativnega raziskovalnega projekta Advanced Ray-tracing Techniques in Radio Environment (krajše ART), pridobljenega na razpisu Javne agencije za raziskovalno dejavnost Republike Slovenije (krajše ARRS). Namen projekta je karakterizirati radijsko okolje in izboljšati radijsko lokalizacijo s pomočjo naprednih tehnik sledenja žarkom.

V naslednjih poglavjih bomo podrobneje opisali namestitev, uporabo in izvedbo sistema, ki nudi programski dostop do algoritmov za vzporedno simuliranje propagacije radijskih valov po prostoru, skladno z zahtevami aktivnosti A3.1, A3.2, A3.3 in A3.4. V poročilo programskega sklopa A2.2 smo že vključili kratka navodila za uporabo, katera smo v tem dokumentu dopolnili in prilagodili glede na spremembe, ugotovljene v času programskega sklopa A2.4. V zadnjih poglavjih bomo s konkretnimi meritvami in rezultati potrdili pričakovane prednosti oblačne infrastrukture in SAAS sistema ART, ki omogoča razširjeno uporabo tako na področju HPC storitev in pohitrenega reševanja stojno zahtevnih problemov, kot tudi hkratnega procesiranja in streženja večjega števila zahtevkov uporabnikov.

V okviru programskega sklopa A2.4 smo testirali integrirane algoritme, preučili odražanje in izkoristek elastične infrastrukture glede na različne testne primere, poenostavili splošno uporabo sistema in izdelavo simulacijskih modelov ter radijskih okolij. Predvidene spremembe posameznih komponent smo dodajali ali odstranjevali iterativno glede na ugotovljene pomanjkljivosti ali izboljšave, katere smo določili z rezultati vsakokratnih izvajanj testnih primerov.

Poleg razširjenega testiranja in izboljšav splošnega delovanja smo izpopolnili in poenostavili uporabniški dostop do rešitev s pomočjo popolne integracije API-ja v 3D modelirno orodje Blender. Na ta način smo omogočili pridobivanje rezultatov dinamičnih animiranih simulacij, ki nudijo vpogled v radijsko karakteristiko okolja pri premikajočih izvorih ali objektih. Pridobljene prednosti smo izkoristili in predstavili v sklopu sodelovanja z evropskim projektom MANTIS, kjer smo simulirali jakost sprejemnega signala senzorjev obremenitve pogonske gredi v stiskalnicah za pločevino. Izsledki omenjene raziskave so dokumentirani v knjigi *The MANTIS Book – Cyber Physical System Based Proactive Collaborative Maintenance*. Poleg uporabniku prijaznejšega oddaljenega dostopa izpopolnjen sistem izboljšuje varnost pred vdori in zagotavlja večjo stabilnost s pomočjo preverjanja vhodnih parametrov.

2 Uporabniška navodila

Uporabniška navodila so namenjena sistemskim administratorjem, razvijalcem programske opreme ali raziskovalcem, ki želijo uporabljati sistem ART. V prvem poglavju so opisani postopki namestitve sistema, preostali del pa se osredotoča na izgradnjo simulacijskega modela in uporabo aplikacijskega vmesnika.

2.1 Namestitev sistema

Oblačne ali porazdeljene storitve se izvajajo na gruči medsebojno povezanih naprav. Za nameščanje, konfiguriranje ter zaganjanje storitev in podpornih sistemov to pomeni zamudno večkratno izvajanje podobnih operacij v ukazni lupini, kar lahko privede do napak in nekonsistentnosti. V izogib tem posledicam in zaradi hitrejšega dela poznamo orodja, s pomočjo katerih v nekem programskem jeziku definiramo želeno stanje sistema.

Eno izmed orodij za nameščanje oblačnih storitev, primerno za potrebe projekta ART, je Ansible. Ansible uporablja datotečni format YAML za opis stanja nameščenih storitev in knjižnic sistema, katerega uveljavlja s pomočjo protokola SSH (ang. Secure Shell) za oddaljen dostop. Ta protokol je dostopen in splošno uporabljen pri nameščanju oblačnih storitev, kar omogoča namestitev sistema s pomočjo orodja Ansible brez predhodnih ročnih posegov v oddaljeno infrastrukturo.

Sistem ART sestavljajo jedrne komponente tipa Manager, Worker in Delegator ter storitve tretjih oseb, kot so RabbitMQ, PostgreSQL in NFS. Jedrene komponente so razvite v programskem jeziku Javascript z uporabo orodja NodeJS, ki za delovanje zahteva ustrezno zagonsko programsko okolje.

Implementacija namestitve sistema ART se nahaja v Git repozitoriju pod imenom ‘project-art’ in vsebuje kopico datotek, katerih pomen in strukturo bomo razložili v nadaljevanju. Razlaga datotek je potrebna tako za razumevanja delovanje sistema kot tudi za predstavitev možnosti prilagajanja na dano okolje (ang. environment). Za boljše razumevanje delovanja se priporoča predhodno poznavanje konceptov namestitvenega orodja Ansible, za samo uporabo pa zadošča razumevanje sintakse opisa okolja.

Namestitev sistema ART z orodjem ansible zaženemo z ukazom

```
ansible-playbook -i environment/<ime-okolja>/inventory playbook.yml
```

Primer 1: Zagon Ansible namestitve

v osnovni mapi projekta. Vrednost za opcijo '-i' omogoča izbiro namestitvenega okolja na način, da uporabnik poda pot do tekstovne datoteke 'inventory', ki seznam imen naprav, njihove IP naslove ter prijavne podatke za SSH dostop. Tako navedene oblačne ali fizične naprave pripadajo skupinam (ang. groups), navedenem v oglatem oklepaju, kot je razvidno iz spodnjega primera namestitvene datoteke 'inventory'.

Datoteka playbook.yml napravam ali skupinam naprav določa vloge in tako posredno predpiše pravila, katere programske komponente naj se namestijo v določeno napravo in kako naj se skonfigurirajo. Namestitveni sistem ART pozna več skupin naprav (manager, worker, delegator, broker in storage). Naprave iz skupine 'manager' prevzamejo vloge koordiniranja simulacij in so zadolžene za delitev poslov na večje število nalog in enakomerno porazdelitev le teh na naprave iz skupine 'worker', ki prevzamejo vloge procesiranja simulacije in agregacijo delnih rezultatov. Naprave 'delegator' nudijo aplikacijski vmesnik in orodje za avtentikacijo in avtorizacijo uporabnikov. Skupina 'broker' označuje naprave, ki skrbijo za medsebojno komunikacijo ostalih naprav, 'storage' pa skrbi za dolgotrajno hrambo vhodnih podatkov in končnih rezultatov.

Omenjene storitve, oziroma v kontekstu namestitve skupine in vloge, v večini nimajo posebnih strojnih zahtev. Izjema sta v našem primeru edino podatkovna baza, ki za nemoteno delovanje zahteva vsaj 4GB prostega pomnilnika, in storitve tipa 'worker', ki poganjajo zahtevne operacije sledenja žarkov in posledično zaradi pohitrenega delovanja zahtevajo dostop do pospešenih grafičnih procesorjev in podpore knjižnice CUDA verzije 8. Primeri opisov oblačne infrastrukture s pomočjo datoteke 'inventory' se nahajajo v mapi 'environment' pod istoimensko mapo okolja. Spodaj je prikazana vzorčna vsebina datoteke 'inventory' osmih naprav, kjer vsaka vsebuje IP naslov, vrata storitve SSH, uporabniški račun za nameščanjeter pot do zasebnega RSA ključa na lokalnem računalniku za SSH dostop.

```
[delegator]
art-delegator ansible_host=10.10.54.101 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
[database]
art-database ansible_host=10.10.54.5 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
[manager]
art-manager ansible_host=10.10.54.10 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
[broker]
art-broker ansible_host=10.10.54.20 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
[storage]
art-storage ansible_host=10.10.54.30 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
[workers]
```

```
art-worker1 ansible_host=10.10.54.41 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
art-worker2 ansible_host=10.10.54.42 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
art-worker3 ansible_host=10.10.54.43 ansible_port=6033 ansible_user=root
    ansible_private_key_file=~/credentials/art/art_keys/id_rsa
```

Primer 2: Primer namestitvene datoteke 'inventory'.

Za namestitev na testno okolje je omenjena datoteka že priložena. V primeru posodobitve sistema ali ponovne namestitve potrebujemo namestitveno orodje Ansible in ključ RSA za vzpostavitev povezave SSH. Če želimo samo posodobiti obstoječe okolje s posodobljenimi komponentami, ne pa ponovno konfigurirati celotnega sistema, lahko poženemo zgoraj omenjeni namestitveni ukaz z dodano oznako (ang. tag) 'pullandupdate', kot je prikazano spodaj.

```
ansible-playbook -i environment/<ime-okolja>/inventory --tags=pullandupdate playbook.yml
```

Primer 3: Zagon Ansible namestitve

Za hitro testiranje in spoznavanje sistema je najbolj prikladno okolje Vagrant, ki vzpostavi in namesti gručo virtualnih naprav na lokalno napravo. Pogoj za rabo Vagranta je, da gostiteljev operacijski sistem in primarni virtualizacijski sistem podpirata rabo GPU-ja znotraj virtualne naprave, v kar se v tem dokumentu ne bomo spuščali in bomo privzeli, da najeta infrastruktura že ima urejen dostop in ustreza strojnim zahtevam.

Poleg direktnega zagona orodja Ansible za namestitev programskega sistema projekt nudi namestitev z uporabo orodja Make, ki posredno požene zgoraj omenjeni ukaz in predhodno poskrbi za vzpostavitev virtualnih naprav v okolju Vagrant. Na ta način je se odpira možnost za razširitve, ki preko API-ja ponudnika infrastrukture avtomatično rezervirajo in vzpostavijo želena strojna sredstva, preden orodje Ansible prične konfigurirati programsko okolje in nameščati sistem ART.

Orodje Make se uporablja preko ukazne lupine tako, da se administrator postavi v osnovno mapo namestitvenega projekta, poimensko definira okoljsko spremenljivko v ukazni lupini 'ENV' z želenim imenom okolja in izvede enega ali več od spodaj navedenih ukazov.

```
make all
make create
make delete
make provision
```

Primer 4: Zagon Make namestitve

Ukaz ‘make all’ izvede operaciji ‘create’ in ‘provision’. Prva vzpostavi virtualno infrastrukturo Vagrant, druga pa s pomočjo orodja Ansible namesti sistem ART na novonastale virtualne naprave. Ukaz ‘make delete’ ustavi vse naprave, sprosti rezervirana strojna sredstva in počisti uporabnikov sistem.

Zgoraj omenjene operacije zakrivajo kompleksnejši opis namestitve, ki se nahaja v posameznih datotekah in mapah. Na osnovnem nivoju se nahaja datoteka ‘playbook.yml’, ki definira vloge prej omenjenih skupin naprav. Del datoteke, ki definira skupino ’manager’, je podan v naslednjem primeru.

```
- name: Provision an ART manager
  hosts: manager
  become: yes
  become_user: root
  vars_files:
    - env_vars/{{ env }}.yml
  roles:
    - core
    - nfs
    - rabbitmq
    - manager
```

Primer 5: Definicija vlog skupine ‘manager’

Kot je razvidno iz tega izpisa, imamo za vsako komponento svojo vlogo. Imena vlog oziroma map s pripadajočimi definicijami razkrivajo dejanske uporabljene storitve, ki so na nivoju inventarja zakrite pod generičnimi imeni. To omogoča lažje razširitve ali zamenjavo posamezne storitve z enakovredno alternativo. Kot primer lahko vzamemo podatkovno bazo, ki je na nivoju vlog definirana kot Postgre.

Druga pomembna mapa, poimenovana ‘env_vars’, vsebuje splošne nastavitve sistema ART, katerih ni potrebno spreminjati pri migraciji med ponudniki oblačnih storitev. V njej so določena uporabniška imena in gesla za dostop do podatkovne baze, posrednika, lokacija NFS datotečnega sistema z vstopnimi točkami ter vrata, na katerih bodo dostopne posamezne storitve.

```
|-- ansible.cfg
|-- environments
|   |-- arctur
|   |   |-- inventory
|   |-- vagrant
|       |-- inventory
|       |-- manager
|       |   |-- Vagrantfile
|       |-- worker
|           |-- Vagrantfile
|-- env\_vars
|   |-- _all.yml
|-- roles
|   |-- delegator
|   |-- manager
```

```
|   |-- worker
|   |-- postgre
|   |-- core
|   |-- cuda
|   |-- nfs
|   |-- rabbitmq
|-- Makefile
|-- playbook.yml
```

Primer 6: Struktura skript za nameščanje sistema

Mapa 'environments' vsebuje podmape, ki definirajo parametre za namestitev sistema na določeno strojno okolje, kot je že bilo omenjeno na začetku tega poglavja. Trenutno namestitveni projekt nudi prilagojene nastavitve za okolje Arctur in Vagrant. Slednje je namenjeno vzpostavitvi razvojnega okolja in testiranju funkcionalnosti sistema, zaradi sposobnosti vzpostavitve gruče navideznih naprav na eni fizični napravi.

Za podrobno razumevanje namestitve sistema je ključno poznavanje vsebine map, kjer se nahajajo definicije že prej omenjenih vlog (ang. roles). Spodaj je prikazana struktura vloge 'manager', ki na osnovnem nivoju vsebuje tri mape.

V podrobnosti nameščanja programske opreme tretjih oseb, kot so RabbitMQ, NFS, PostgreSQL, CUDA, OptiX in specifik operacijskega sistema CentOS, se ne bomo spuščali, ker sistem ART ne zahteva posebno prilagojenih nastavitev delovanja. Za te komponente zadošča že osnovna vzpostavitev in omogočen dostop do storitev, kar je opisano v dokumentacijah omenjenih orodjih. Smiselno je omeniti, da se te storitve namestijo ob pogonu Ansibla orodja avtomatično na naprave, ki pripadajo skupinam, za katere so te vloge vključene v datoteki 'playbook.yml'.

```
|-- defaults
|   |-- main.yml
|-- tasks
|   |-- git_repos.yml
|   |-- main.yml
|-- templates
    |-- config_manager.json.j2
    |-- pm2_manager.json.j2
```

Primer 7: Struktura datotek za namestitev komponente

Prva mapa na seznamu, poimenovana 'defaults', vsebuje datoteke, ki definirajo privzete vrednosti nastavitev in se uporabijo, če istoimenski parametri niso definirani v prej omenjeni mapi splošnih nastavitev. V mapah 'tasks' vsake vloge se nahajajo namestitvene skripte vseh razvitih in odvisnih komponent sistema. Datoteke vsebujejo poimenske naloge, označene s pomišljaji, in se izvršujejo v enakem redu kot so navedene. Namestitvene naloge, ki se pojavljajo v namestitvah komponent, zaobjemajo konfiguracijo

varnostnih podsistemov operacijskega sistema in požarnih zidov. Temu sledi povezovanje na repozitorij programske kode Git, prenos programske kode na ciljno napravo, vzpostavitev vstopnih map do datotečnega sistema NFS ter konfiguracija storitve in zagon. Spodaj je prikazana strnjena sekvenca namestitve vloge 'worker', ki za razliko od ostalih razvitih komponent vsebuje korak namestitve RTX algoritmov, ne vsebuje pa konfiguracije varnostnih mehanizmov, ker naprava ni direktno dostopna iz drugih omrežij.

```
- name: Create the NFS mount directory.
- name: Unmount the NFS directory if it has been mounted before.
- name: Mount the NFS directory.
- name: Setup the Git repos
- name: Create algorithm build directory.
- name: Configure with cmake.
- name: Compile the raytracer binary.
- name: Install npm packages
- name: Set service configuration
- name: Set pm2 configuration
- name: Run service under pm2.
```

Primer 8: Struktura skript za namestitev komponente

Namestitvene skripte komponent namenoma ne vsebujejo namestitev odvisnih storitev ali knjižnic. Te so vsebovane v ločenih vlogah, od katerih je za zgoraj opisano sekvenco pomembna vloga imenovana 'core', katera vsebuje namestitve dodatnih repozitorijev 'epel' operacijskega sistema CoreOS, na katerih se nahajajo zahtevana orodja za uspešno namestitev komponent. Dodatno vloga poskrbi za namestitev orodja Git in NodeJS, brez katerih ni mogoče prenesti izvorne kode in pognati storitev na ciljni napravi. Posledično naprave iz skupin 'manager', 'delegator' in 'worker' prevzemajo vloge posameznih komponent in vlogo 'core', kot je razvidno iz zgoraj prikazanega odseka datoteke 'playbook.yml'.

Po prenosu izvorne kode vsake komponente in pred zagonom je potrebno pravilno izpolniti nastavitve, ki se nahajajo v datoteki 'config/config.json' v osnovni mapi pripadajočih projektov. Predloga nastavitev z vsemi podprtimi parametri se nahaja na isti lokaciji v datoteki 'default.json'. Ta korak namestitve, tako kot vsi ostali, je avtomatiziran s pomočjo orodja Ansible in s pripadajočo funkcionalnostjo izpolnjevanja in prenašanja predlog nastavitev, ki se nahajajo v mapi 'roles/art/templates' namestitvenega projekta. Te vsebujejo namesto posameznih konfiguracijskih vrednosti spremenljivke, navedene v zavitih oklepajih, in služijo kot reference do parametrov orodja Ansible.

```
{
    "rabbitmq": {
        "url": "{{ manager_int_ip }}",
```

```
        "port": 5672,
        "user": "{{ rabbitmq_user }}",
        "password": "{{ rabbitmq_password }}",
        "timeout": 1000,
        "prefetch": 10,
    },
    "db": {
        "url": "{{ manager_int_ip }}",
        "port": 5432,
        "dialect": "postgres",
        "database": "{{ database }}",
        "user": "{{ database_user }}",
        "pass": "{{ database_password }}"
    },
    "splitJobs": false,
    "logLevel": "info"
}
```

Primer 9: Ansible predloga nastavitev komponente Manager

S parametri orodja Ansible smo se že srečali pri opisu splošnih nastavitev, datoteke z inventarjem naprav in datoteke s privzetimi nastavitvami posamezne vloge. Globalni parametri orodja Ansible so definirani v mapi namestitvenega projekta 'env_vars' in vseh datotekah, vsebovanih v mapah z imenom 'defaults'. Ansible podpira navajanje parametrov tudi na drugih lokacijah, vendar smo se zaradi lažjega razumevanja pri razvoju sistema omejili samo na omenjene.

```
---
  - remote_user: root
  - delegator_url: "art.xlab.si"
  - delegator_api_port: 80
  - delegator_ssh_port: 6033
  - manager_ssh_port: 6033
  - worker_ssh_port: 6034
  - nfs_host_dir: "/raytracer"
  - nfs_guest_dir: "/tracer"
  - rabbitmq_user: raytracer
  - rabbitmq_password: sledenjezarkom
  - database: "art"
  - database_user: "art"
  - database_password: "art"
```

Primer 10: Primer Ansible nastavitev

Poleg uporabe tako definiranih parametrov Ansible omogoča referenciranje vrednosti posameznih naprav, podanih v datoteki 'inventory'. Z refrenciranjem IP naslovov, navedenih v datoteki 'inventory', in navajanjem le teh v predlogah, se poenoti njihova uporaba in izogne večkratnemu ponavljanju. To uporabniku ali sistemskemu administratorju nudi poenostavljen prenos namestitve iz enega v drugo okolje na način, da prilagodi ali na novo definira datoteko 'inventory'.

Pred namestitvijo posamezne konfiguracijske datoteke namestitveno orodje vsako predlogo prebere in zamenja spremenljivke z njihovimi vrednostmi. Predloga tako postane veljavna konfiguracija storitve za izbrano okolje in se po prenosu izvorne kode namesti na prej omenjeno ciljno mesto.

Po uspešno izvedeni namestitvi posamezne komponente sledi namestitev in konfiguracija orodja 'pm2', ki skrbi za poganjanje procesov. Ob morebitnih napakah, sesutju storitev ali ponovnem zagonu naprave omogoča avtomatiziran ponovni zagon in vzpostavitev prvotnega stanja procesov. Poleg tega omogoča lažji način pregledovanja, nadzora porabe sredstev in stanja posamezne ART storitve. Da lahko 'pm2' požene in tako spremlja življenjski cikel procesa, potrebuje nastavitveno datoteko, ki vsebuje pot do zagonske datoteke. Namestitveni projekt vsebuje predlogo omenjene datoteke za vsako komponento. Te se nahajajo v mapah 'templates' vsake vloge, zraven predlog nastavitev komponent.

2.2 Konfiguracija sistema

Po uspešni namestitvi sistema in pred splošno javno uporabo je priporočljivo ponastaviti določene nastavitve iz varnostnih razlogov. Če uporabljamo sistem ART samo v zaprtem okolju, te niso potrebne in lahko nemudoma začnemo uporabljati vse funkcionalnosti.

Najpomembnejša varnostna prilagoditev po namestitvi je sprememba gesla administratorskega računa, ki je privzeto nastavljen na uporabnika 'admin' z geslom 'admin'. Uporabniško ime in geslo je mogoče prilagoditi po prvi uspešni prijavi in pridobitvi avtentikacijskega žetona na način, da se ustvari novi uporabnik z administratorskimi pravicami in onemogoči privzeti administratorski račun. Podroben opis zahtevkov in odgovorov je prikazan v poglavju 3.1. Postopek zamenjave poteka tako, da administrator najprej izvrši zahtevek za pridobitev žetona. Tega uporabi kasneje pri vseh ostalih zahtevkih tako, da ga priloži glavi HTTP protokola. Z zahtevkom za kreiranje uporabnika ustvari nov račun z želenim imenom in geslom in atributom 'role' z vrednostjo 'admin'. Po uspešnem odgovoru izvrši zahtevek za pridobitev novega žetona z uporabo novih avtentikacijskih podatkov. V tem trenutku sta v sistemu zabeležena dva uporabnika z administratorskimi pravicami. Na novo prijavljeni administrator izvrši zahtevek za spremembo podatkov privzetega uporabnika z namenom, da onemogoči njegovo uporabo. To stori z uporabo zahtevka za spremembo nastavitev posameznega uporabnika, ki trenutno sprejeme samo en atribut 'enabled' z vrednostmi 'true' ali 'false', ki določa aktivnost računa.

Druga nastavitev določa privzete lastnosti materialov simuliranih objektov in njihove šifrante, določene z identifikacijsko številko materiala in pri-

padajoče skupine. Namestitvene skripte za lažji začetek uporabe sistema predhodno naložijo osnovne materiale, ki se lahko uporabijo pri izdelavi in kasnejši simulaciji propagacije radijskih valov in so prikazani na seznamu spodaj. Za pregled podrobnih fizikalnih lastnosti posamezne snovi lahko uporabnik izvrši zahtevek za pridobitev ali posodobitev celotne datoteke v XML formatu, kot je podrobneje opisano v poglavju **??**. V poglavju 2.5 bomo obrazložili uporabo spodaj navedenih identifikatorjev pri izgradnji simulacijskega modela.

```
<materials>
        <group id="0">
                <description> Concrete </description>
                <material id="0"> <description> Heavy </description> </material>
                <material id="1"> <description> Medium </description> </material>
                <material id="2"> <description> Light </description> </material>
                <material id="3"> <description> Siporex </description> </material>
                <material id="4"> <description> Cement </description> </material>
        </group>
        <group id="1">
                <description> Stone </description>
                <material id="0"> <description> Brick </description> </material>
                <material id="1"> <description> Clinker </description> </material>
                <material id="2"> <description> Lime-Brick </description> </material>
        </group>
        <group id="2">
                <description> Plaster Board </description>
                <material id="0"> <description> Heavy </description> </material>
                <material id="1"> <description> Medium </description> </material>
                <material id="2"> <description> Light </description> </material>
        </group>
        <group id="3">
                <description> Wood </description>
                <material id="0"> <description> Heavy </description> </material>
                <material id="1"> <description> Medium </description> </material>
                <material id="2"> <description> Light </description> </material>
                <material id="3"> <description> Particle Board </description> </material>
                <material id="4"> <description> Chip Board </description> </material>
                <material id="5"> <description> Cottonwood </description> </material>
                <material id="6"> <description> Birch </description> </material>
                <material id="7"> <description> Fir </description> </material>
        </group>
        <group id="4">
                <description> Glass </description>
                <material id="0"> <description> Regular </description> </material>
                <material id="1"> <description> Mirror </description> </material>
                <material id="2"> <description> Jena </description> </material>
                <material id="3"> <description> Triple </description> </material>
        </group>
        <group id="5">
                <description> Plastic </description>
                <material id="0"> <description> Glass Fiber </description> </material>
                <material id="1"> <description> Polystyrene </description> </material>
                <material id="2"> <description> Perspex </description> </material>
                <material id="3"> <description> Polyester Resin </description> </material>
                <material id="4"> <description> Polyglass </description> </material>
                <material id="5"> <description> Natural Rubber </description> </material>
                <material id="6"> <description> Silicone Rubber </description> </material>
        </group>
```

```
        <group id="6">
                <description> Others </description>
                <material id="0"> <description> Metal </description> </material>
                <material id="1"> <description> Sand </description> </material>
                <material id="2"> <description> Clay </description> </material>
                <material id="3"> <description> Paper </description> </material>
                <material id="4"> <description> Asphalt </description> </material>
                <material id="5"> <description> Earth </description> </material>
        </group>
</materials>
```

Primer 11: Seznam nabora vgrajenih fizikalnih lastnosti materialov

2.3 Programski dostop do storitev

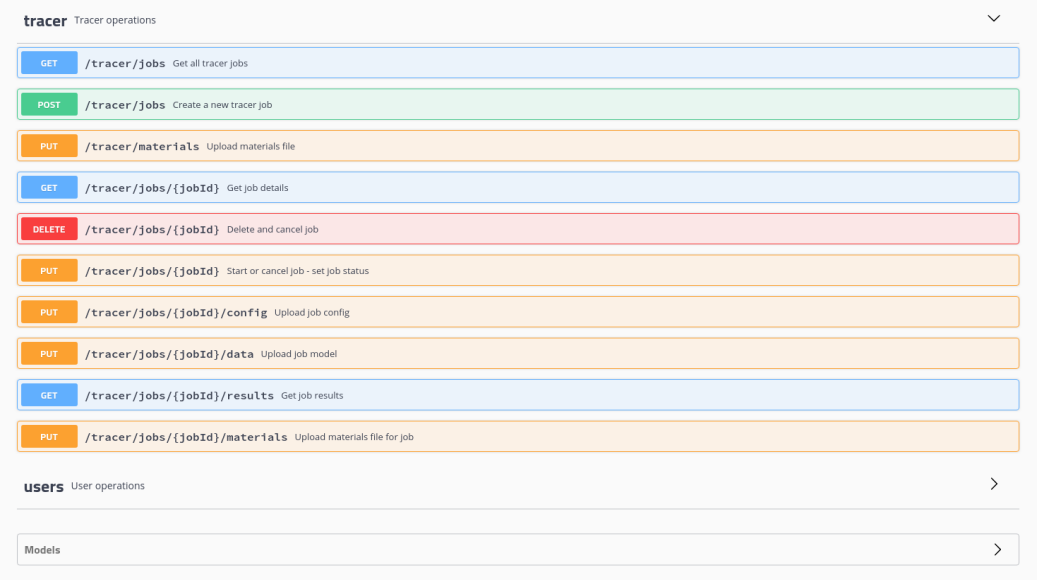
V tem poglavju bomo na splošno opisali uporabo, časovni potek zahtevkov ter pričakovane rezultate. Strukturo posameznih zahtevkov, obliko njegovih vhodnih podatkov in odgovora pa podrobno opisuje poglavje 3.

Komponenta Delegator ima vlogo aplikacijskega strežnika in služi kot vstopna točka v sistem preko dobro definiranega REST aplikacijskega vmesnika. Ta je implementiran s pomočjo orodja Swagger in nudi možnost abstraktnega opisa končnih vstopnih točk (ang. end-points), vhodnih modelov in rezultatov, ki skupaj definirajo aplikacijski protokol. Abstrakten opis protokola omogoča enostaven prenos v drugo razvojno okolje in hitro spremembo podatkovnega formata (JSON, XML, ...). Poleg omenjenih prednosti uporabniku nudi spletno stran z dokumentacijo aplikacijskega vmesnika in orodje za hitro izvrševanje zahtevkov.

Aplikacijski vmesnik je dostopen na mrežnem naslovu komponente Delegator, kot je bilo določeno v postopku namestitve. Uporabnik dostopa do spletnega vmesnika preko brskalnika na naslovu,

```
http://<delegator\_hostname>/docs
```

kjer se mu prikaže pregled podprtih končnih točk ter shem vhodnih in izhodnih modelov aplikacijskega vmesnika. Ob kliku na posamezen razdelek se odprejo podrobne informacije o vhodnih parametrih, vzorčni model zahtevka ter tipi napak, ki se ob nepravilni uporabi lahko vrnejo namesto pričakovanega odgovora. Prikazane zahtevke lahko uporabnik tudi izvršuje in po želji prilagaja predlogo vsebine. S klikom na tipko 'Try it out' se spremeni pogled izbranega zahtevka v način urejanja, kar spremeni prikazano predlogo zahtevka v urejevalno tekstovno polje. Ker shema aplikacijskega protokola temelji na formatu JSON formatu, je potrebno nivoju API-ja urejati vhodne modele v skladu s tako definirano sintakso, ki je prilagojena strojni obdelavi in razvoju grafičnih vmesnikov.



Slika 1: Prikaz grafičnega vmesnika za delo s posli

V grobem zahtevke delimo v tri skupine: zahtevke za delo z uporabniškimi računi, zahtevke za upravljanje materialov in splošnih nastavitev sistema ter zahtevke za upravljanje poslov.

Prvo skupino zahtevkov smo delno že opisali v poglavju 2.2, ko smo govorili o ustvarjanju novega administratorskega računa. Funkcionalnost ni omejena samo na prej omenjene tipe uporabnikov, ampak omogoča tudi preostalim uporabnikom spremembo prijavnega gesla kot je obrazloženo v razdelku 3.1. Tako kot administratorski uporabnik tudi ostali potrebujejo avtentikacijski žeton za upravljanje poslov, katerega dodajo v HTTP glavo vsakega zahtevka.

Tretja skupina zahtevkov obsega kreiranje in spremljanje uporabniških poslov, pošiljanje modelov, postavitev oddajnih anten in receptorskih ravnin ter pregled končnih rezultatov posamezne simulacije. Postopek se prične s proženjem zahtevka za kreiranju novega posla, kot je opisano v poglavju 3.3. Zahtevek vrne identifikacijsko kodo, s katero preko ostalih v nadaljevanju opisanih zahtevkov konfiguriramo, poženemo, spremljamo potek in pridobivamo rezultate.

Kreiranju zahtevka sledi karakterizacija radijske okolice in receptorskih točk. Zahtevek mora vsebovati parametre oddajne antene, nastavitve tipa simulacije, želeno natančnost oziroma zahtevnost izvajanja ter sprejemno ravnino ali točko za izračun pulznega odziva. Podroben opis zahtevka in vsebine se nahaja v poglavju 3.4.

Uspešni karakterizaciji okolice sledi zahtevek, ki mora vsebovati posebno

strukturiran in parametriziran geometrijski model, kakršnega razumejo RTX algoritmi za izvajanje sledenja žarkov. Strukturiran je v XML formatu in vsebuje opis geometrije, ki jo sestavljajo posamezni zidovi in luknje. Zid ali luknja v modelu je šeststrano telo, podano z šestimi štirikotnimi površinami. Vsako površino določa šest točk oziroma dva trikotnika. Podrobne informacije o zgradbi modela in izvedbi zahtevka se nahajajo v poglavju 3.4.

Pred zagonom simulacijskega posla ima uporabnik možnost podajanja fizikalnih lastnosti materialov, ki jih uporablja v geometrijskem modelu. Na ta način se lahko namesto privzetih materialov, ki jih sme posodabljati samo administrator, poda posebej izdelan nabor, ki je vezan samo na dotični posel. Podrobnosti zahtevka se nahajajo v poglavju 3.6.

Po uspešni konfiguraciji in namestitvi vseh potrebnih podatkov sledijo zahtevki za zagon in upravljanje poslov simulacije. Uporabnik lahko po uspešnem zagonu spremlja napredek ali pridobi informacije o morebitnih napakah. Podrobnosti posameznih zahtevkov za zagon in pridobivanje statusa se nahajajo v poglavju 3.3.

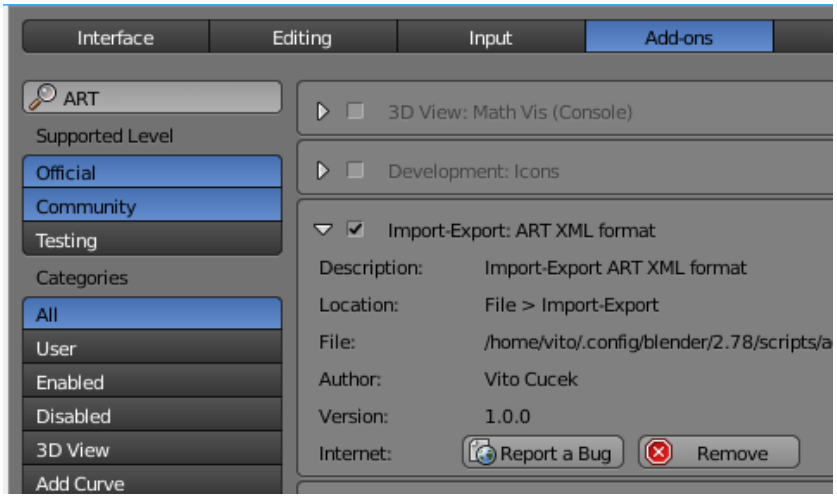
Uspešnemu zaključku simulacijskega posla sledi pridobivanje rezultatov. Sama vsebina rezultatov je odvisna od karakterizacije radijske okolice in nastavitev simulacijskega posla. Podrobne informacije o nastavitvah RTX simulacijskih algoritmov se nahajajo v poglavju 3.9, oblika in uporaba zahtevkov pa v poglavju 3.8.

2.4 Delo z grafičnim vmesnikom

Za lažjo uporabo in testiranje razvite rešitve smo izdelali grafični vmesnik. Izdelan je v programskem jeziku Python in se uporablja kot vtičnik odprtokodnega programa Blender. Že v času programskega sklopa A2.2 smo izdelali sorodno rešitev, ki je ločeno od sistema ART omogočala izdelavo vhodnih modelov in prikaz rezultatov simulacij, tokrat pa smo funkcionalnost razširili z direktnimi klici na programski vmesnik. Na ta način lahko končni uporabnik uporablja večino funkcij sistema ART direktno preko grafičnega vmesnika programa Blender.

Vtičniki se nahajajo v projektu ‘srv-calc-algorithm’ v mapi z imenom ‘blender’. Ta vsebuje dve podmapi, imenovani ‘io-scene-art’ in ‘render-art’. Prva vsebuje vtičnik za uvoz rezultatov in izvoz modelov ter karakteristik oddajnika, sprejemnikov in splošnih nastavitev simulacijskih algoritmov. Druga mapa vsebuje vtičnik, ki preko REST API-ja komunicira z oddaljenim sistemom ART. Vtičnika sta kompatibilna z različico Blender-ja 2.78.

Namestitev vtičnika poteka tako, da prej omenjene mape (vsako zase) zapakiramo skupaj z vsebino v istoimensko ZIP datoteko. V tej obliki je možno vtičnik preko grafičnega vmesnika namesti v Blender s klikom na



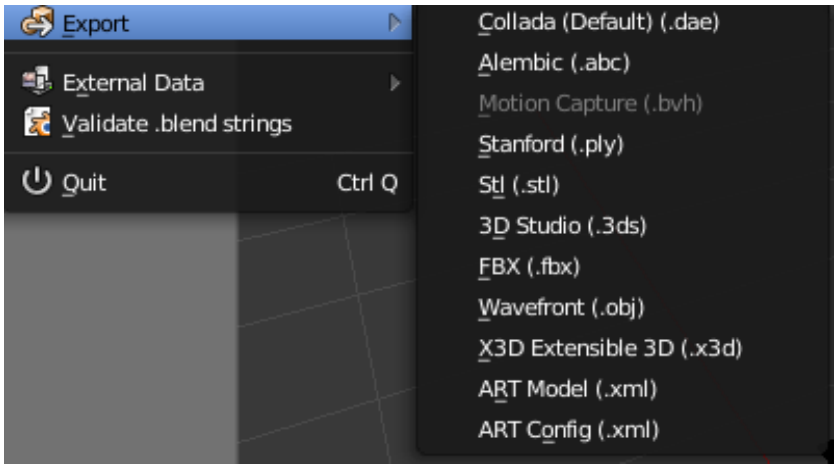
Slika 2: Prikaz aktivacije vtičnika za uvoz in izvoz modelov

meni v zgornjem levem kotu okna aplikacije 'File->User Preferences...'. Prikaže se novo okno z nastavitvami programa in seznamom trenutno registriranih vtičnikov. S klikom na tipko 'Install from file...' se odpre okno za izbiro datoteke. V datotečnem sistemu poiščemo prej izdelano ZIP datoteko, jo označimo in potrdimo izbiro s tipko 'Install from file...'. Vtičnik se tako namesti na sistem. in sicer v uporabnikovo mapo splošnih nastavitev programa Blender, ki se na operacijskem sistemu Linux nahaja v mapi ' /.config/blender/2.78/scripts/addons/'.

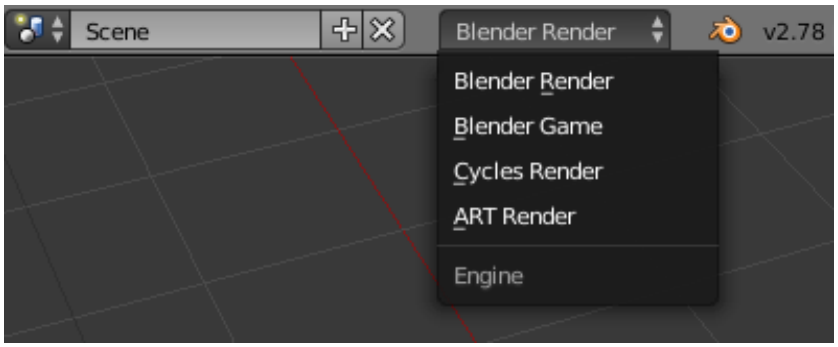
Po namestitvi je potrebno vtičnik še aktivirati. To storimo tako, da poimensko poiščemo vsakega v seznamu registriranih in ga aktiviramo s klikom na potrditveno polje (ang. check box). Če ob ponovnem zagonu programa želimo imeti na voljo aktivirane vtičnike, moramo nastavitve shraniti s klikom na tipko 'Save User Settings'.

Po uspešni namestitvi se v grafični vmesnik dodajo nove komponente za delo in interakcijo z oddaljenim sistemom. Med možnostmi za uvoz in izvoz modelov tako lahko najdemo tri nova polja, poimenovana 'ART', 'ART Model' in 'ART Config'. Na sliki 3 lahko vidimo izbirni polji za izvoz parametriziranega modela in splošnih nastavitev simulacijskih algoritmov.

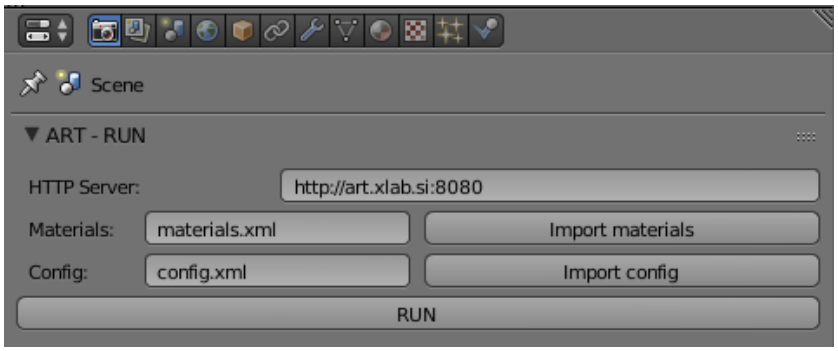
Druga funkcionalnost je 'ART render', ki se nahaja zraven izbire 'Blender in Cycles' v glavnem meniju programa. Ta preklopi način delovanja programa tako, da prilagodi običajne možnosti izrisa scene v povezljiv način ART simulacije. Na sliki 4 je prikazana plošča za izbiro oddaljenega sistema in tipka za zagon simulacije. Po preklopu delovanja v način 'ART Render' način uporabnik začne modelirati geometrijsko okolje, katerega postopek je opisan v poglavju 2.5.



Slika 3: Prikaz menija za izvoz modelov in karakteristik okolja



Slika 4: Izbirni meni za preklop med načini delovanja



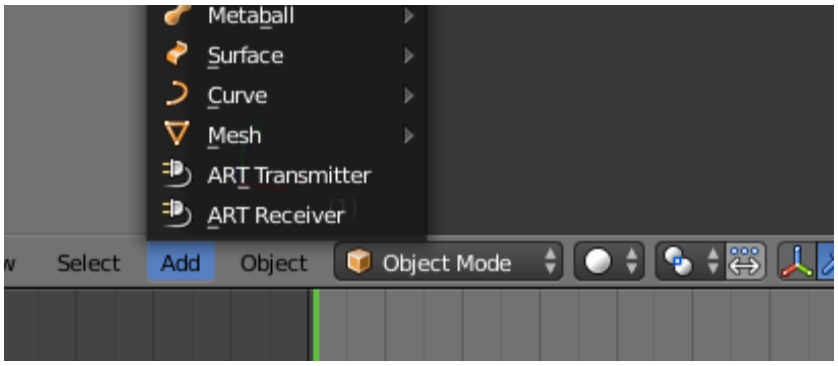
Slika 5: Grafična komponenta za upravljanje simulacijskih poslov

Na koncu ali med samim modeliranjem ima uporabnik možnost zagnati simulacijo in preučiti rezultate. To stori tako, da na skrajno desni plošči programa v meniju izbere zavihek 'Render' ter v polje 'HTTP Server' vpiše URL, na katerem so dostopne končne točke sistema ART. URL mora vsebovati samo protokol (http), ime strežnika in vrata. Rezultati simulacije se čez čas prikažejo direktno v 3D pogledu programa na mestu, kjer je bila definirana receptorska ravnina. Za konfiguracijo simulacije lahko uporabnik tudi vnese v polje 'Materials' in 'Config' poti do datoteki s konfiguracijo. V primeru da sta polji prazni, se bodo za simulacijo uporabljali privzete nastavitve. Za podroben prikaz rezultatov lahko uporabnik uporabi pogled 'UV/Image Editor', ki prikaže rezultat kot zaslonsko sliko, katero je mogoče tudi izvoziti v poljubnem formatu s klikom na izbiro 'Image->Save...'.

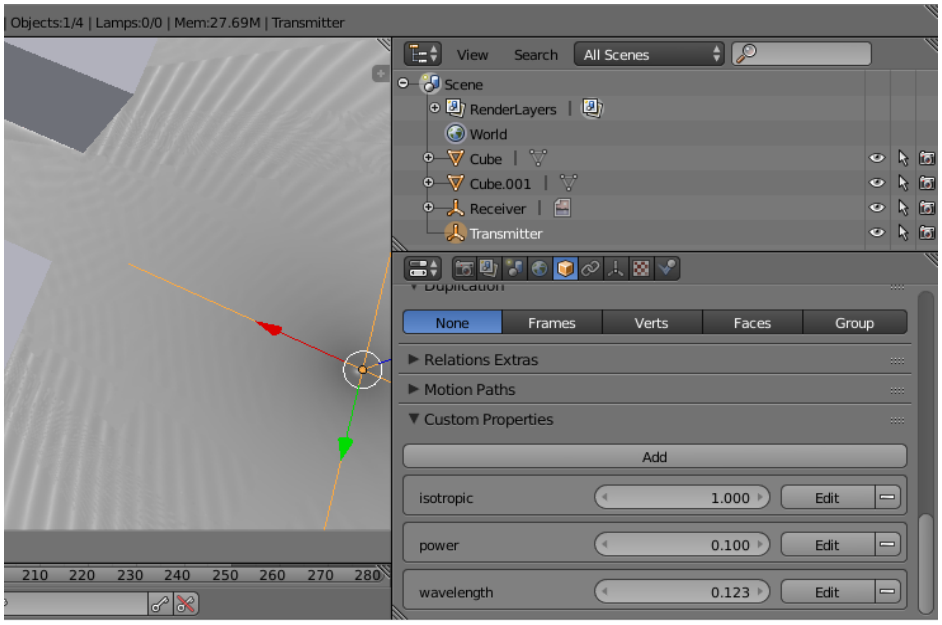
2.5 Izdelava grafičnih modelov in simulacijskega okolja

Vtičnik 'io_scene_art' omogoča izgradnjo geometrijskega modela in pretvorbo obstoječega modela, zapisanega v enem izmed podprtih standardnih formatov, v parametriziran geometrijski model za simulacijo. Poleg specifičnih lastnosti posameznih objektov algoritem za simulacijo radijskih valov zahteva opis geometrije na nivoju zidov, lukenj, ravnin in konveksnih robov, kar onemogoča neposredno uporabo standardnih zapisov (OBJ, 3DX, FBX, ...), ki opisujejo geometrijo na nivoju trikotnikov. Vtičnik za izvoz na podlagi soležnih trikotnikov in materialov samodejno preračuna in preoblikuje model z uporabo naprednih orodji za urejanje in analizo geometrijskih mrež, vgrajenih v samo jedro Blender-ja.

Postopek izdelave modela se prične z uporabo standardnih orodji za 3D modeliranje ali pa z uvozom že obstoječega. Temu sledi kreiranje oddajne antene in receptorske ravnine s klikom na meni 'Add' v 3D pogledu, kot je prikazano na sliki 6.



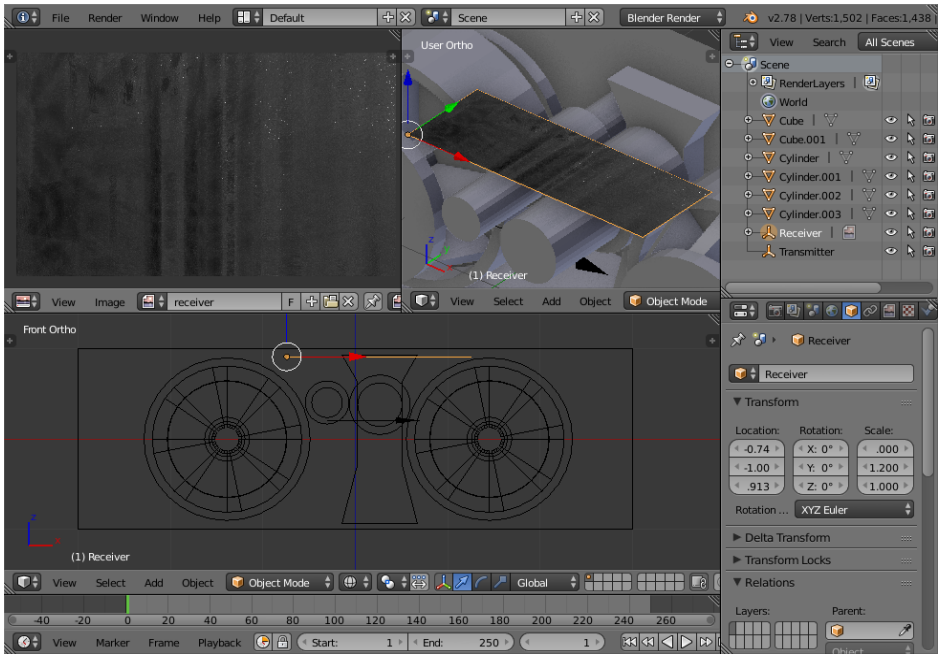
Slika 6: Prikaz menija za kreiranje oddajne antene in receptorske ravnine



Slika 7: Prikaz nastavitev oddajnika

Fizikalni podatki o oddajni anteni in receptorski ravnini se vnašajo s pomočjo dodatnih parametrov (ang. Custom Parameters) preko Blender vmesnika, kot je prikazano na sliki 7 v spodnjem desnem kotu. Na sliki je prikazan primer vnosa prametrov 'isotropic', 'power' in 'wavelength' oddajne antene. Objekt receptorske ravnine podpira parametra 'ppm' in 'isotropic'. Obrazložitev pomena parametrov skupaj z fizikalnimi enotami se nahaja v poglavju 3.9. Na enak način kot konfiguriramo oddajno anteno in receptorsko ravnino, podajamo tudi materiale posameznim objektom simulacije, z navajanjem parametrov 'mat_group' in 'mat'. Prva vrednost prejme identifikacijsko številko skupine matriala, druga pa identifikacijsko številko materiala v izbrani skupini. System ART ima že ob zagonu vgrajenih nekaj osnovnih materialov in pripadajočih fizikalnih lastnosti, ki so navedeni v poglavju 2.2.

Po končanem urejanju lahko uporabnik izvozi sceno s klikom na 'File->Export->ART(.xml)', izbere želeno mapo in vnese ime datoteke. Pridobljeno datoteko uporabi pri zahtevku aplikacijskega vmesnika za pošiljanje geometrijskega modela (put /api/tracer/task/:id/data). Preprostejši način, ki je bil razvit v sklopu zadnje aktivnosti, pa omogoča direktno proženje aplikacijskega vmesnika sistema ART, kot je opisano v prejšnjem poglavju.



Slika 8: Prikaz izgub radijskega signala

2.6 Pridobivanje in pregled rezultatov simulacije

Po uspešno pridobljenih rezultatih simulacije lahko uporabnik s pomočjo aplikacije Blender prikaže pridobljene vrednosti receptorskih sfer v obliki sivin. Omenjen prikaz deluje samo pri rezultatih, ki vsebujejo izgubo signala v decibelih. Za uvoz rezultatov uporabnik v aplikaciji izbere 'File->Import->ART(.xml)'. Primer uspešno uvožene receptorske ravnine z rezultati o izgubah signala je prikazan na sliki 8. Zgornji levi del Blenderjevega vmesnika v tem primeru prikazuje povečano sliko receptorske ravnine, zgornji srednji del pa receptorsko ravnino, umeščeno v sam model. V prikazanem primeru gre za model stiskalnice za pločevino, v katerem receptorska ravnina predstavlja možne lokacije sprejemne antene. Oddajna antena senzorja obremenitve pogonske gredi je prikazana kot črna puščica na spodnjem delu slike.

3 Zasnova aplikacijskega vmesnika

Aplikacijski vmesnik je zasnovan na arhitekturi REST, protokolom HTTP in podatkovnem formatu JSON. Značilnost REST arhitekture je, da na neodvisen način povezuje strežniški del in odjemalce z izmenjavo reprezentativnih objektov, ki ponazarjajo želeno ali trenutno stanje sistema. Uporabnikova seja se hrani izključno na hrani samo na odjemalcu, kar omogoča neodvisno obravnavo posameznih zahtevkov in posledično večjo skalabilnost sistema.

HTTP protokol se največkrat uporablja v povezavi z REST arhitekturo, ker na enostaven in najbolj razširjen način omogoča izgradnjo večslojnih sistemov. Tako lahko storitev, na katero je naslovljen zahtevek, in odjemalec komunicirata brez predhodne konfiguracijepreko vrste posrednikov, ki preusmerjajo, analizirajo ali prevajajo pretok podatkov z namenom hitrejšega distribuiranega izvajanja, vzdrževanja, zagotavljanja razpoložljivosti, združljivosti ali drugih razlogov. HTTP protokol definira možne operacije (GET, POST, PUT, DELETE...), stanje povezave in končne poti virov (ang. endpoints), katere določi posamezna storitev. Skupaj z definicijo podatkovnega modela tvori aplikacijski protokol, preko katerega se izmenjuje stanje sistema in proži oddaljene procedure.

Za format za izmenjevanje informacij smo izbrali JSON (ang. JavaScript Object Notation) zaradi preproste berljivosti in podpore v različnih programskih jezikih in orodjih za gradnjo spletnih storitev. Spletni strežnik je izdelan v programskem jeziku JavaScript in temelji na orodjih NodeJS in Express. Aplikacijski vmesnik je definiran s pomočjo urejevalnika Swagger, ki nudi preprost pregled nad možnimi operacijami preko spletne strani. Podatkovni model je možno po potrebi spreminjati brez večjih posegov v sistem, ker so vhodni in izhodni parametri programsko definirani v neodvisni datoteki v formatu YAML. Ta datoteka opisuje končne vstopne točke zahtevkov in podatkovni format, na podlagi katerega orodje nudi validacijo zahtevkov.

V spodnjih poglavjih so opisane funkcionalnosti sistema ART, dostopne prek aplikacijskega vmesnika. Sledijo si v logičnem zaporedju, od registracije uporabnika, prijave v sistem, konfiguracije in pošiljanja nalog do pridobivanja končnih rezultatov simulacije.

3.1 Prijava v sistem in upravljanje uporabnikov

Dodajanje in odstranjevanje uporabnikov ter pridobivanje seznama vseh uporabnikov je dovoljeno samo uporabnikom z administratorskimi pravicami.

GET /api/users — Pridobi registrirane uporabnike

```
Odgovor:
{
  "users": [
    "40d1c452-43e5-11e8-842f-0ed5f89f718b",
    "40d1c844-43e5-11e8-842f-0ed5f89f718b",
    "40d1cb1e-43e5-11e8-842f-0ed5f89f718b"
  ]
}
```

Zahtevek vrne seznam indentifikatorjev vseh registriranih uporabnikov. Identifikator se uporablja pri izbrisu ali spremembi gesla in imena uporabnika.

Z naslednjim zahtevkom lahko administrator ustvari novega uporabnika. Atribut 'role' določa poblastila uporabnika (admin ali user). Uporabnik s pooblastili user nima možnosti pregleda poslov in pridobivanja rezultatov dugih uporabnikov, ustvarjanja ali brisanja uporabniških računov in spreminjanja splošnih nastavitev sistema.

POST /api/users — Ustvari novega uporabnika

```
Zahtevek:
{
  "username": "uporabnik5",
  "password": "*********",
  "role": "admin" // admin/user
}

Odgovor:
{
  "id": "40d1cb1e-43e5-11e8-842f-0ed5f89f718b",
}
```

Pirjavljen uporabnik lahko nastavi ali pridobi svoje ime.

GET /api/users/:id — Pridobi podatke uporabnika

```
Odgovor:
{
```

```
  "username": "uporabnik5",
  "name": "Uporabnik",
}
```

PUT /api/users/:id — Posodobi podatke uporabnika

```
Zahtevek:
{
  "name": "Uporabnik",
}
```

Prijavljen uporabnik lahko spremeni svoje uporabniško geslo.

PUT /api/users/:id/auth — Posodobi uporabniško geslo

```
Zahtevek:
{
  "username": "uporabnik5",
  "olsPassword": "*********",
  "newPassword": "*********",
}
```

Administrator lahko izvede izbris uporabnika iz sistema na podlagi uporabnikove identifikacijske kode. Prijavljen uporabnik lahko izbriše svoj uporabniški račun na način, da pod navedeni ID uporabnika v poti zahtevka navede 'self'.

DELETE /api/users/:id — Administrator izbriše uporabnika

DELETE /api/users/self — Uporabnik izbriše samega sebe

Pred začetkom uporabe aplikacijskega vmesnika je potrebno izvesti zahtevek za avtorizacijo z uporabniškim imenom in geslom. Zahtevek v odgovoru vrne avtentikacijski žeton (ang. token).

POST /api/uaa/user/auth/token — Ustvari uporabniški žeton

```
Zahtevek:
{
  "username": "uporabnik5",
  "password": "*********",
}
```

```
Odgovor:
{
  "accessToken": "lx5189i64817y752xy74104x32305tt32x50349x5x79f1xj34xxkxp51",
  "refreshToken": "938o9u0xx447gx4t421h513c4x474813m2065w4w3xx523xqq006xrr21",
}
```

Avtorizacijski žeton je potrebno predložiti v glavi vsakega HTTP zahtevka, kot je prikazano na primeru spodaj.

```
GET /api/tracer/jobs HTTP/1.1
Host: art.xlab.si
Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l
User-Agent: curl/7.58.0
Accept: */*
```

Primer 12: Primer HTTP glave zahtevka z avtentikacijskim žetonom

Uporabniški žeton je veljaven največ en dan. Po poteku žetona je potrebno poslati zahtevek za pridobitev novega z uporabo obnovitvenega žetona (ang. refresh token).

POST /api/uaa/user/auth/refresh Obnovi uporabniški žeton

```
Zahtevek:
{
  "refreshToken": "938o9u0xx447gx4t421h513c4x474813m2065w4w3xx523xqq006xrr21",
}

Odgovor:
{
  "accessToken": "lx5189i64817y752xy74104x32305tt32x50349x5x79f1xj34xxkxp51",
  "refreshToken": "938o9u0xx447gx4t421h513c4x474813m2065w4w3xx523xqq006xrr21",
}
```

3.2 Nastavitve sistema

Administrator ima možnost dodajanja in odstranjevanja privzetih materialov ter spreminjanja fizikalnih lastnosti obstoječih. Zahtevek podpira podajanje podatkov tako v XML kot v JSON formatu. Spodaj je prikazana definicija zahtevka in podatkovnega modela. Podrobne informacije z opisi posameznih vrednosti in fizikalnih enot se nahajajo v poglavju 3.9.

PUT /api/tracer/materials — Posodobi privzete materiale

```
Zahtevek:
{
 "frequency": 1800,
 "group": [
   {
     "id": 0,
     "description": "Concrete",
     "material": [
       {
         "id": 0,
         "description": "Heavy",
         "rel_permittivity": 9,
         "rel_permeability": 1,
         "conductivity": 0.09
       },
       {
         "id": 1,
         "description": "Medium",
         "rel_permittivity": 6,
         "rel_permeability": 1,
         "conductivity": 0.07
       }
     ]
   },
   {
     "id": 1,
     "description": "Stone",
     "material": [
       {
         "id": 0,
         "description": "Brick",
         "rel_permittivity": 4,
         "rel_permeability": 1,
         "conductivity": 0.04
       }
     ]
   }
 ]
}

Odgovor (HTTP response code):
   201: Materiali uspešno naloženi
   400: Napačna oblika zahtevka
   500: Nepričakovana napaka
```

3.3 Upravljanje poslov

Odjemalec ustvari posel tako, da poda ime posla in opis. Zahtevek vrne identifikacijsko kodo, ki se nanaša na novonastali posel in se uporablja pri konfiguraciji posla, pošiljanju geometrijskega modela, brisanju in poizvedbi rezultatov.

POST /api/tracer/jobs — Ustvari novi posel

```
Zahtevek:
{
  "name": "Simulacija A14",
  "description": "Simulacija časovnih zakasnitev radijskega oddajnika 3201244004"
}

Odgovor:
{
  "jobId": "20b9ea36-449a-11e8-842f-0ed5f89f718b"
}
```

Administrator za razliko od ostalih uporabnikov pridobi vse posle.

GET /api/tracer/jobs — Pridobi posle

```
Odgovor:[
 "5206ddcb-60a7-4c3d-9caf-49b4ec5c52df",
 "a67ca7ac-f4f5-4c50-ab4c-9d676152cd9f",
 "96798c72-c108-44b3-81b4-3c45d9398b5d",
 "826160e3-a14a-4f6d-b1ed-7856db24feea",
 "6ea006bb-e0b3-415c-b287-eda6627a5d71",
 "01112744-c8f1-4c57-93e4-7acf16242f14",
 "bf5e2844-c91f-40de-862d-326031074b83"
]
```

Rezultat zgornje zahteve je seznam vseh identifikacijskih kod poslov. Za pridobitev vseh podrobnosti o nekem specifičnem poslu z znano identifikacijsko kodo se lahko uporabi naslednji zahtevek.

GET /api/tracer/jobs/:jobId — Pridobi vse podatke posla

```
Odgovor: {
 "id": 7,
 "jobId": "bf5e2844-c91f-40de-862d-326031074b83",
 "userId": "art",
 "name": "",
 "description": "",
 "config": null,
 "model": null,
 "material": null,
 "state": "created",
 "deleted": false,
 "createdAt": "2019-03-07T16:18:23.477Z",
 "updatedAt": "2019-03-07T16:18:23.477Z"
}
```

3.4 Določanje geometrijskega modela

Odjemalec mora podati geometrijski model, na katerem bo algoritem tekel, v obliki JSON ali XML. Pogoj je, da je posel že ustvarjen in njegova identifikacijska koda znana. Spodaj je prikazan primer zahtevka.

PUT /api/tracer/jobs/:jobId/data — Določi geometrijski model posla *jobId*

```
Zahtevek:
{
  "architecture": {
    "scene": {
      "wall": [
        {
          "material": "2 0",
          "texture": 0,
          "tex_height": 1,
          "dimension": "5.00000000 5.00000000 5.00000000",
          "rotation": "-0.00000000 0.00000000 6.28318531",
          "translation": "13.52000000 4.17000000 0.00000000",
          "separating_floors": 0,
          "floor": 0,
          "surfaces": [
            {
              "id": 0,
              "point": [
                "5.00000000 0.00000000 5.00000000",
                "0.00000000 0.00000000 0.00000000",
                "5.00000000 0.00000000 0.00000000",
                "0.00000000 0.00000000 0.00000000",
                "5.00000000 0.00000000 5.00000000",
                "0.00000000 0.00000000 5.00000000"
              ]
            },
            .
            .
            .
          ]
        }
      ]
    }
  }
}

Odgovor (HTTP response code):
   201: Geometrijski model uspešno naložen
   400: Napačna oblika zahtevka
   404: Posel s podanim jobId ne obstaja
   500: Nepričakovana napaka
```

3.5 Konfiguracija algoritma

Podobno kot geometrijski model mora odjemalec tudi konfiguracijo algoritma podati v obliki JSON ali XML. Konfiguracija opiše sprejemnike oziroma re-

ceptorsko ravnino, oddajnik in razne parametre za sledenje žarkom. Spodaj je prikazan primer zahtevka. Podroben opis vseh parametrov je podan na koncu poglavja, v razdelku 3.9.

PUT /api/tracer/jobs/:jobId/config — Ustvari konfiguracijo posla

Zahtevek:

```
{
  "version": "3.0",
  "request_id": 1420751223,
  "description": "description",
  "transmitter": {
    "wavelength": 0.1225,
    "position": "0.00000000 2.65000000 2.50000000",
    "direction": "0.00000000 0.00000000 1.00000000",
    "isotropic": 0,
    "power": 0.1
  },
  "receivers": {
    "area": {
      "dimension": "13.00000000 13.00000000",
      "rotation": "0.00000000 0.00000000 0.00000000",
      "translation": "-9.50000000 -5.83333333 2.50000000",
      "ppm": 30
    },
    "position": "0.00000000 2.65000000 2.50000000",
    "direction": "0.00000000 0.00000000 1.00000000",
    "isotropic": 0
  },
  "method": 0,
  "subdivision": {
    "depth": 10,
    "rays_per_step": 40962
  },
  "raytracing": {
    "depth": 1,
    "accumulate": 7,
    "rx_radius": 0.015,
    "diffraction": {
      "edge_radius": 4.145608029883936
    },
    "bloom": {
      "k": 14,
      "m": 288
    },
    "power_limit": {
      "max_amp": "3.909700e-009",
      "max_d_to_a": "1.599300e+004"
    },
    "cir_entries": 1
  }
}
```

Odgovor (HTTP response code):

```
  201: Konfiguracija algoritma za posel uspešno naložena
  400: Napačna oblika zahtevka
  404: Posel s podanim jobId ne obstaja
  500: Nepričakovana napaka
```

3.6 Fizikalne lastnosti materialov

Odjemalec lahko, če ne želi uporabljati privzetih materialov, kakršne je določil administrator, doda v posel svoje materiale. Struktura zahtevka je identična tisti, ki je opisana v razdelku 3.2, razlikuje se le v URLju. Spodaj so prikazani možni odgovori na zahtevek.

PUT /api/jobs/:jobId/materials Ustvari materiale posla z identifikacijsko kodo jobId

```
Odgovor (HTTP response code):
   201: Materiali za podani posel uspešno naloženi
   400: Napačna oblika zahtevka
   404: Posel s podanim jobId ne obstaja
   500: Nepričakovana napaka
```

3.7 Status posla

Po končani konfiguraciji posla ga odjemalec lahko zažene ali že zagnani posel prekine.

PUT /api/tracer/jobs/:jobid Spremeni status posla

```
 Zahtevek:
 {
   "status:"start"
 }
 // V primeru, da želimo posel prekiniti, lahko uporabimo "status":"cancel"

Odgovor (HTTP response code):
   201: Status posla uspešno nastavljen
   400: Napačna oblika zahtevka
   404: Posel s podanim jobId ne obstaja
   500: Nepričakovana napaka
```

Posle je možno tudi izbrisati, kar izbriše iz sistema vse podatke posla ter ga v primeru, da posel že teče, tudi prekine.

DELETE /api/tracer/jobs/:jobId — Prekini in izbriši posel

```
Odgovor (HTTP response code):
   204: Posel uspešno izbrisan
   404: Posel s podanim jobId ne obstaja
   500: Nepričakovana napaka
```

3.8 Pridobivanje in interpretacija rezultatov

Za pridobitev rezultatov posla se uporabi zahtevek spodaj. Odgovor zahtevka je le v XML formatu.

GET /api/tracer/jobs/:jobId/results — Pridobi rezultate posla

```
Odgovor:
       <?xml version="1.0" encoding="UTF-8"?>
       <signal>
               <request_id> 1420751223 </request_id>
               <preproc_time> -1.00000000 </preproc_time>
               <gpu_time> -1.00000000 </gpu_time>
               <loss_db>
                       <num_x> 390 </num_x>
                       <num_y> 390 </num_y>
                       <rx>
                               <y>
                                       <x> 58.00053766 </x>
                                       <x> 57.98354419 </x>
                                       <x> 57.96654476 </x>
                                               .
                                               .
                                               .
                                       <x> 1000 </x>
                               </y>
                                .
                                .
                                .
                       </rx>
               </loss_db>
       </signal>
```

3.9 Opisi parametrov zahtevkov

V tem razdelku navajamo vse parametre za konfiguracijo algoritma z zahtevkom, dokumentiranim v razdelku 3.5.

transmitter — object

Opis vseh podrobnosti oddajnika. Vsebuje polja, opisana spodaj: wavelength (valovna dolžina), position (položaj oddajne antene), direction (usmerjenost oddajne antene) antene, isotropic (izotropnost oddajne antene) in power (moč oddajanja).

wavelength — number [m]

Valovna dolžina oddajanja v metrih. Vrednost ne sme močno preseči valovne dolžine materialov (0.18~m).

position — string

Absolutni položaj antene v prostoru. Podan mora biti kot trojica x, y in z koordinat, ločenih s presledki. Vrednosti naj bodo vneš ene v metrih.

direction — string

Orientacija anteninega dipola. Podan mora biti kot enontni vektor s tremi elementi (x, y in z), ločenimi s presledki.
Če je zahtevana izotropična antena, potem smer nima pomena.

isotropic — int32

Antene so oblikovane tako, da enakomerno porazdelijo moč oddajanja v vse smeri, če je zastavica isotropic postavljena (ima vrednost 1). Če je vrednost isotropic enaka 0, se antene obnašajo kot idealni dipoli. Ostale vrednosti niso dovoljene.

Primer:

```
{
   "isotropic": 0
}
```

power — number [W]

Moč oddajnika v vatih. Parameter je prisoten le zaradi združljivosti, saj ga sledilec žarkom ne uporablja.

receivers — object

Opisuje opazovano ravnino, sestavljeno iz sprejemnih anten ali v nadaljevanju točk za vzorčenje signala.
Vsebuje polja, opisana spodaj: area (receptorska ravnina), position (pozicija centroida receptorske površine), direction (usmerjenost sprejemnih anten) in isotropic (izotropnost sprejemnih anten).
Orientacija receptorske antene in vzorec oddajanja velja za vse vzorčne točke. Opazovana ravnina ima obliko pravokotnika z določeno gostoto vzorčnih točk.

area — object

Definicija opazovane signalne površine in gostote sprejemnikov na tej ravnini, na kateri se signali evalvirajo.
Vsebuje polja, opisana spodaj: dimension (velikost ravnine), rotation (usmerjenost ravnine), translation (položaj ravnine), ppm (gostota evalvacije).

dimension — string

Dimenzija je dvojica števil, ki pomenijo velikost receptorske ravnine v metrih.

rotation — string

Rotacija receptorske ravnine mora biti podana kot trojica s presledki ločenih števil, ki predstavljajo Eulerjeve kote v radianih.

translation — string

Položaj vektorske ravnine, podan kot trojica s presledki ločenih koordinat x, y, z v metrih.

ppm — int32 [m]

Točke na meter (ang. Points per meter) opisuje gostoto točk na opazovani ravnini.
Gostota je podana v eni dimenziji, t.j. 10 točk na meter pomeni 100 točk na kvadratni meter.
Če je vrednost nastavljena na 0 se izračunava pulzni odziv oddajne antene v točki pozicije receptorske ravnine.
Resolucijo pulznega diagrama določa parameter cir_entries.

position — string

Točka na opazovani ravnini. Podana mora biti kot trojica x,y in z koordinat, ločenih s presledki. Vrednosti naj bodo vnešene v metrih.

method — int32

Simulacijska metoda, ki naj se uporabi. V trenutni različici je podprta le vrednost 0, ki pomeni metodo SBR (Shooting and bouncing rays).

subdivision — object

Element subdivision določa število vseh žarkov, ki jih bo oddajnik oddal.
Vsebuje podelementa depth (globina) in rays_per_step (žarki na korak), opisana spodaj.

depth — int32

Element depth pod elementom subdivision določa rekurzivno stopnjo delitve mreže ikozaedra.
Pri dani globini n je skupna količina izsevanih valovnih front enaka 10*22n+2.

rays_per_step — int32

Število žarkov, ki se jim sledi v posameznem koraku ima vlogo optimizacijskega parametra.
Porazdelitev obremenitve GPUjev se zanaša na ta parameter.

Uporabiti je potrebno vrednost, ki da najkrajši čas izvajanja glede na specifikacije uporabljenih grafičnih kartic.

raytracing — object

Vsebuje več spodaj opisanih parametrov za algoritem sledenja žarkom.

depth — int32

Določa najvišje število zaporednih interakcij (reflekcije, refrakcije, difrakcije) med sledenjem posamičnemu žarku.

accumulate — int32

Bitna maska, ki določa uporabljene tipe žarkov na opazovanih točkah:
1-direktni valovi, 2-odbiti valovi, 4-lomljeni valov, 8-uklonjeni valovi.
Tipični vrednosti bitne maske sta 7 in 15.

rx_radius — number [m]

Radij recepcijskih krogel za detekcijo žarkov, podan v metrih. Krogle morajo imeti neničelne radije, da se zagotovi vsaj en zadetek na valovno fronto.
Primerna vrednost je odvisna od vrednosti števila vseh žarkov, globine sledenja žarkov in velikosti geometrije okolja.

diffraction — object

Vsebuje parametre, pomembne za difrakcijske dogodke. Trenutno se lahko nastavi le radij kotnih cilindrov.

edge_radius — number

Difrakcija je modelirana po uniformi teoriji. Vsak žarek, ki potuje blizu difrakcijskega kota, sproži nove žarke znotraj Kellerjeve cone.

Za detekcijo proženja difrakcijskih žarkov se uporabi podoben pristop kakor pri recepcijskih kroglah. Velikost edge_radius se mora podrejati številu prvotnih žarkov,
globini sledenja in razsežnosti geometrije okolja.

bloom object

Definicija Bloomovega filtra za izogibanje dvojnemu štetju.

k int32

Število neodvisnih zgoščevalnih (ang. hash) funkcij, ki se uporabljajo za Bloomove filtre.

m int32

Število bitov za shranjevanje bloom filtru. Vrednost naj bo več kratnik števila 32.

power_limit object

Vsebuje spodaj opisana parametra za omejevanje sledenja žarkom do sprejemljive izgube signala.

max_amp string

Določa mejo izgube signala, pri kateri se sledenje žarkov ustavi.

max_d_to_a string

Druga konstanta, ki se uporablja za ustavitev žarkov.
Da nam spremeljivo razmerje dolžine in absorbcije.

cir_entries int32

Če je vrednost elementa 1, je izguba signala evaluirana na več točkah oziroma na receptorski ravnini.
Če je vrednost večja od ena se simulira pulzni odziv oddajnika v to točki središča receptorske ravnine.
Glej position parameter receptorske ravnine.

4 Metrike sistema in ugotovitve

Obstoječi algoritem smo distribuirali na način, kot je opisan v poročilu programskega sklopa A2.2. Izbrani način bomo ponovno na kratko povzeli v naslednjem odstavku.

Algoritem sledenja žarkom porazdelimo na način, da vsakemu sledilniku povemo skupno število instanc ter zaporedno številko trenutne. Iz celotne geometrije modela, sestavljenega iz indeksiranih površin, ter podanih podatkov o instancah, vsak sledilnik na podlagi skupne formule določi njemu dodeljene površine. Formula za izračun uporabi zaporedno identifikacijsko številko površine in jo pretvori v modul števila razpoložljivih sledilnikov. Tako pridobljena vrednost se med izvajanjem algoritma primerja z zaporedno številko instance sledilnika in uporabi kot filter za izločanje valovnih front prvega reda. To pomeni, da vsak sledilnik obravnava samo valovne fronte, ki zadenejo pripadajoče površine ali nastanejo kot posledica trka. V primeru, ko površina pripada trenutnemu procesu sledilnika, se ob interakciji z njo ustvarijo nove valovne fronte, ki predstavljajo odboj, disperzijo ali uklon elektromagnetnega valovanja. Če izračunana vrednost formule ni enaka zapredni številki trenutne instance, se valovna fronta pred interakcijo s površino zavrže, ker bo obravnavana na drugi. Po zaključku vseh procesov tipa sledilnik dobimo delne disjunktne rezultate, katere lahko z majhno porabo strojnih virov (v primerjavi s preostalim algoritmom) seštejemo soležne vrednosti in dobimo končni rezultat.

V idealnem primeru lahko trajanje simulacije, oziroma pohitritev izvajanja definiramo kot:

$$T = max(\frac{t_1}{m_s}, \frac{t_2}{m_s}, \frac{t_3}{m_s}, ..., \frac{t_n}{m_s}) + k \quad (1)$$

Vrednosti $t_0, t_1, t_2...$ predstavljajo čas izvajanja celotne simulacije na posamezni mašini, m predstavlja število instanc ter k čas potreben za agregacijo delnih rezultatov. V praksi se zgodi, da čas izvajanja simulacije na posamezni instanci presega idealni delež vseh. Vzrok je neenakomerna porazdeljenost valovnih front med posamezne površine, oziroma razlike v direktni izpostavljenosti elektromagnetnemu sevanju. To se najbolj izraža pri modelih, ki imajo manjše število večjih površin direktno izpostavljenih oddajni anteni, kar ima za posledico da večje breme procesiranja prevzame manjši delež instancac sledilnika. Pri obsežnih modelih s podrobno geometrijo in odnosno večjim številom površin, verjetnost za isti pojav statistično pada. Ker algoritem deli simulacijski posel na podlagi direktno izpostavljenih površin oddajni anteni, se v nadaljni obravnavi osredotočamo zgolj na te.

Za lažje razumevanje se omejimo na primer, kjer infrastrukturo sestavljajo računsko enakovredne mašine. Zgornji izraz se poenostavi na:

$$t_0 = \frac{t}{m_s}, \quad T = t_0 + k \tag{2}$$

Vrednost t predstavlja čas izvajanja celotne simulacije na eni mašini, t_0 pa idealni čas izvajanja algoritma sledenja žarkom na porazdeljenem sistemu. Zaradi statistične verjetnosti, da se površine v direktnem zornem kotu oddajne antene ne porazdelijo enakomerno med instance, lahko iz zgornje enačbe razvijemo enačbo za predviden povprečen čas procesiranja na podlagi variance binomske porazdelitve. Iščemo gostoto verjetnosti, da posamezna instanca procesira primarne valovne fronte n površin od skupno n_s, ki so direktno izpostavljene oddajni anteni. V spodnji enačbi p predstavlja verjetnost, da površina pripada izbrani instanci sledilnika.

$$p = \frac{1}{m_s} \tag{3}$$

Verjetnost, da je izbrani instanci dodeljeno n površin lahko dobimo iz binomske porazdelitve,

$$b(n; n_s, p) = \binom{n_s}{n} p^n (1-p)^{n_s - n} \tag{4}$$

katero lahko aproksimiramo z zvezno na način, da izračunamo povprečno vrednost diskretne in standardni odklon. Zaradi aproksimacije nadaljni popravek časa velja samo kadar je n_s zadosti velik.

$$dP/dn = \mathcal{N}(n_s p,\, n_s p(1-p)) \tag{5}$$

Iz zgornje normalne porazdelitve razberemo varianco s pomočjo katere lahko izračunamo relativen odklon od zgoraj omenjenega idealnega časa.

$$\sigma = \sqrt{n_s p(1-p)} \tag{6}$$

$$\eta = \sqrt{\frac{1-p}{p n_s}} \tag{7}$$

Kot lahko vidimo, se pri večjem številu prej omenjenih površin in fiksnem številu instanc procesorski čas približuje idealni pohitritvi, ker relativen odklon trajanja na posamezni instanci pada. Nasprotno se zgodi, če povečujemo

število instanc pri fiksnem številu površin. V tem primeru se čas procesiranja zmanjšuje dokler je $n_s > m_s$, vendar se vrednost odstopanja postopoma oddaljuje od idealnega časa. Dugače povedano, pri večjem številu instanc in enakem številu direktno obsijanih površin, težje zagotovimo idealno enakomerno porazdelitev.
Celoten čas izvajanja z upoštevanim popravkom opisuje enačna:

$$T = t_0(1 + \sqrt{\frac{m_s - 1}{m_s^2 n_s}}) + k \tag{8}$$

Za potrebe testiranja smo uporabili preprost model kock, oddajnik ter receptorsko ravnino, sestavljeno iz 150 receptorskih sfer, kot je prikazano na sliki 9.

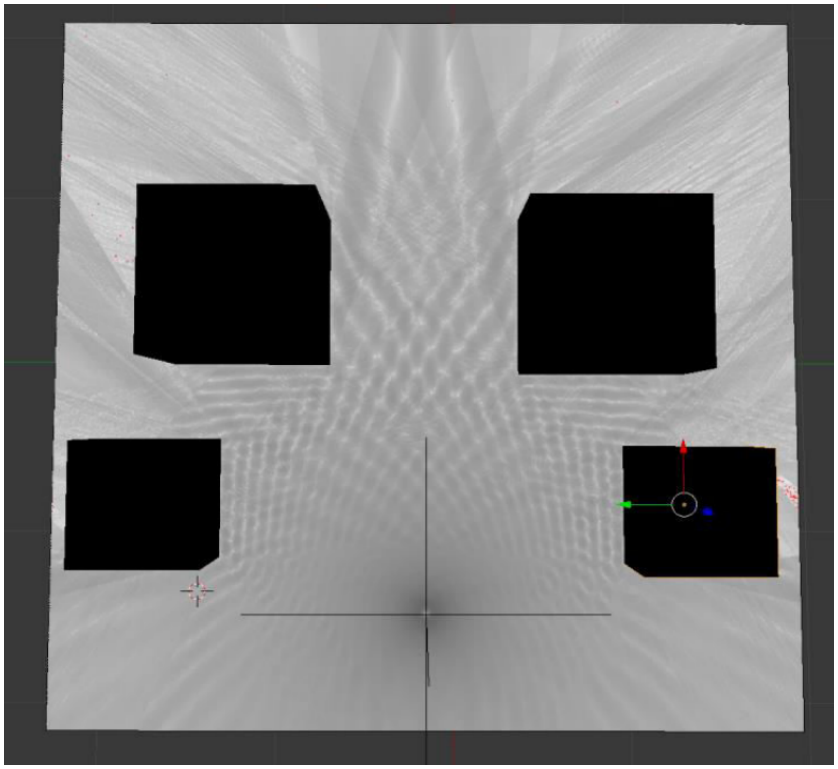
Spodnje meritve se nanašajo na simuliranje izgub radijskega valovanja pri modelu dveh kock:

	$m_s = 1$	$m_s = 2$	$m_s = 3$
Sledilnik 1 $t_1[s]$	1136	240	585
Sledilnik 2 $t_2[s]$	0	923	534
Sledilnik 3 $t_3[s]$	0	0	34
Skupno $t[s]$	1137	1163	1153
Idealen čas $t_0[s]$	1136	568	378
Izračunan čas $T[s]$	1136	769	504
Izmerjen čas $T'[s]$	1137	925	585

Pri simulaciji na dveh mašinah lahko vidimo znatno odstopanje od idelnega časa zaradi neugodne razporeditve površin po sledilnikih. To smo tudi pričakovali, ker ima dani primer samo štiri površne direktno izpostavljene oddajni anteni. V dani situaciji se je naključno zgodilo, da je večina dela bila naložena instanci sledilnika 2.

Spodnji rezultati so pridobljeni pod enakimi pogoji pri simulaciji modela štirih kock:

	$m_s = 1$	$m_s = 2$	$m_s = 3$
Sledilnik 1 $t_1[s]$	4503	923	1686
Sledilnik 2 $t_2[s]$	0	3518	1556
Sledilnik 3 $t_3[s]$	0	0	1304
Skupno $t[s]$	4503	4441	4546
Idealen čas $t_0[s]$	4503	2251	1501
Izračunan čas $T[s]$	4503	2814	1855
Izmerjen čas $T'[s]$	4503	3518	1686



Slika 9: Vzorčni model za testiranje pohitritev distribuiranega izvajanja RTX algoritmov

5 Literatura

Albano M, Jantunen E, Papa G in Zurutuza U (2019). The MANTIS book: cyber physical system based proactive collaborative maintenance. Gistrup; Delft: River Publishers, 2019. https://www.riverpublishers.com/pdf/ebook/RP_E9788793609846.pdf. <18.02.2019>

Coppa E (2017). Hadoop internals. Anatomy of a mapreduce job. http://ercoppa.github.io/HadoopInternals/AnatomyMapReduceJob.html. <18.02.2017>

Northam L, Smits R, Daudjee K in Istead J (2013). Ray tracing in the cloud using mapreduce. 2013 international conference on high performance computing & simulation (HPCS): 19-26. http://ieeexplore.ieee.org.nukweb.nuk.uni-lj.si/document/6641388/. <18.02.2017>

OpenStack. OpenStack Fundation (2017). Introduction to Ironic. https://docs.openstack.org/developer/ironic/deploy/user-guide.html. <18.02.2017>

Shi L, Chen H in Sun J (2009). vCUDA: GPU accelerated high performance computing in virtual machines. 2009 IEEE international symposium on parallel & distributed processing, Papers 8: Patents: 1-11. http://ieeexplore.ieee.org.nukweb.nuk.uni-lj.si/document/5161020/. <18.02.2017>

Shirahata K, Sato H in Matsuoka S (2010). Hybrid map task scheduling on GPU-based heterogeneous clusters. 2010 IEEE second international conference on cloud computing technology and science, Papers 14: 733-40. http://ieeexplore.ieee.org.nukweb.nuk.uni-lj.si/document/5708524/. <18.02.2017>