TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Andrei Voitenko**

# Delta õppehoone keskkonna visualiseerimine

**Bakalaureusetöö (9 EAP)**

Juhendaja: Raimond-Hendrik Tunnel, MSc

Tartu 2018

## Delta õppehoone keskkonna visualiseerimine

**Lühikokkuvõte:**

Käesolev töö kirjeldab Delta õppehoone keskkonna visualisatsiooni loomise protsessi. Töö käigul valmis ühendus Cumulocity platvormi ja visualisatsiooni vahel andmete edastamiseks. Loodi Delta õppehoone simuleeritud JSON formaadis tunniplaan, mille abil oli visualiseeritud igas klassiruumis parasjagu toimuva aine nimetus ning klassiruumi number. Klassiruumi asukoha paremaks arusaamiseks olid loodud sildid, mis tähistavad ruumide numbrid 3D visualisatsioonil. Oli integreeritud ilmastiku visualiseerimine, mis põhineb OpenWeatherMap rakendusliidese andmetel. Integreeriti päikese valguse suunda vastamiseks reaalmaailma päiksele.

**Võtmesõnad:**

Visualisatsioon,

**CERCS:** P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

## Delta building environment visualization

**Abstract:**

This thesis explains how Delta building environment visualization was created. During development, a connection between the Cumulocity platform and the visualization was created to handle data. A mocked schedule in the JSON format was created, which was used to visualise the ongoing subjects in each room. Room labels were placed on each individual room for easier understanding of the room's location. A weather visualization was created, which consists of an integrated OpenWeatherMap application programming interface. Directional light was rotated in the visualization according to real-life sun position and rotation.

**Keywords:**

Visualization, timetable, weather, Cumulocity

**CERCS:** P170 Computer science, numerical analysis, systems, control

# Sisukord

# 1. Sissejuhatus

Hoonetesse sisenedes on oluline, et siseneja teaks kuhu minna ja mis selles majas hetkel toimub. Selleks, et inimesel saaks kohe selgeks, kus asub kabinet või otsitav ruum, on tihtipeale olemas maja plaan. Vaadates seda tekib kujutus, kus inimene asub ja kuhu peab minema. Uue Delta õppehoone ehitamisega, kus hakkavad õppima Tartu Ülikooli arvutiteaduse, majandusteaduse ja matemaatika tudengid, otsustati luua programm, mis annaks kõikidele õppehoone külastajatele informatsiooni majas reaalajas toimuvatest protsessidest.

Ehitatavasse Delta õppehoone ruumidesse on planeeritud erinevaid andureid (temperatuuri, $CO_2$, õhu niiskuse, õhurõhu, õhu kvaliteedi, suitsu, müra tase, valguse tase, liikumise, inimeste loenduri ja kohaloleku andurid), mille andmete põhjal saab arvutada informatsiooni maja, inimeste ja keskkonna kohta. Saadud inimeste arvu informatsiooni põhjal otsustati luua Delta õppehoone visualiseerimise projekt. Visualisatsiooni kasutatakse suure koguse informatsiooni paremaks arusaamiseks [1, 2]. Käesoleva ja A. Nikolajevi töö [3] raames valmib programm, mis aitab inimestel näha vajaliku õpperuumi, vaadata põnevat visualisatsiooni õppehoones toimuvatest tegevustest, klassiruumide tunniplaani ning õppehoone ümbruses olevat ilma.

Fuajeesse planeeritud visualisatsioonis näidatakse 3D hoone mudelit ning selles toimetavaid inimesi. Tudengi Aleksander Nikolajevi paralleelselt tehtav bakalaureusetöö “Delta Building Visualisation and Optimisation” [3] keskendub õppehoone mudeli ja algoritmide integratsioonil. See sisaldab õppehoone mudeli optimeerimist, inimeste mudelite loomist ja nende animeerimist ning inimesi kujutavate mudelite navigatsiooni mudelil. Täpsemalt saab lugeda A. Nikolajevi lõputööst [3].

Käesolev lõputöö jaotub kolme ossa, kus tegeletakse infovahetusega visualisatsiooni ja teiste süsteemide vahel, rakenduse tekstilise lisainformatsiooniga (nt. tunniplaan ja ilm) ning hoonet ümbritseva keskkonna visualiseerimisega.

Peatükis 3 kirjeldatakse käesoleva töö raames tehtud liidestust Cumulocity süsteemiga, millest saadavad maja sensori andmete põhjal leitud inimeste oletatav arv ruumides saadetakse edasi A. Nikolajevi poolt tehtud rakenduse osale inimeste ja tegevuste visualiseerimiseks.

Peatükis 4 kirjeldatakse Tartu Ülikooli Õppeinfosüsteemi põhjal reaalajas igas ruumis toimuva aine üldist informatsiooni (klassiruumi number, aine nimetus, ainele registreeritud tudengite arv) visualiseerimist.

Peatükis 5 kirjeldatakse kolmemõõtmeliste osakeste- ja valgusefektidena ning mudelitena loodud erinevate ilmastikuolude (päikesepaiste, pilved, vihm, lumi, äike) visualisatsioon, mille kohta reaalaja info pärineb ilmaennustuse teenusest.

Delta õppehoone keskkonna visualiseerimisel kasutatakse Unity mängumootorit, kus luuakse skripte C# programmeerimiskeeles, integreeritakse API-d ehk rakendusliidesed ja töödeldakse andmeid. Unity on mängumootor või mängude loomise tööriist mis võimaldab inimestel väga lihtsalt luua arvutimänge [4]. Pärigute koostamiseks kasutatakse süntaksreegleid, mis on näiteks JSON-il. JSON (ing. k. *JavaScript Object Notation*) on lihtsustatud andmevahetustevorming, mis on loodud JavaScripti programmeerimiskeele süntaksi põhjal. JSON andmevahetusvorming on kergesti arusaadav inimestele [5]. Unity mängumootori valik on põhjendatud Unity funktsionaalsusega ja lõputöö autori kogemusega antud mängumootori kasutamises. Alterantiiviks on nt. Unreal Engine 4 (edasi UE4) mängumootori või mõne API nt. OpenGL-i kasutamine.

Võrreldes Unity mängumootoriga on UE4 rakendustel suuremad ressursinõuded arvutile, mängumootorite ressursinõuete võrdlus on toodud tabelis 1. API nt. OpenGL-i kasutamise puuduseks on suurema aja koguse kulutamine mängu või visualisatsiooni loomiseks võrreldes mängumootori kasutamisega [6].

*Tabel 1: Mängumootorite ressursinõuete võrdlus.*

| **Mängumootor** | **OS** | **CPU** | **GPU** | **Mälu** |
|---|---|---|---|---|
| Unity 2017.3[1] | Windows 7 SP1+, 8, 10, 64-bit versions only; Mac OS X 10.9+ | SSE2 toetus | Graafikakaart DX10 (shader mudel 4.0) võimetega | sõltub projekti keerukusest |
| Unreal Engine 4[2] | Windows 7/8 64-bit | Quad-core Intel või AMD, 2.5 GHz või kiirem | Graafikakaart DX11võimetega | 8 GB RAM |

Unity mängumootori skriptide loomiseks kasutatakse kaks programmeerimiskeelt: C# ja JavaScript. Delta õppehoone keskkonna visualiseerimiseks oli valitud C# programmeerimiskeel seoses käesoleva töö autori eelneva kogemusega antud programmeerimiskeeles skriptide loomisel.

Töös kasutatud terminid on toodud lisas I. Visualisatsiooni kompileeritud fail ja selle lähtekood lisas II. Python abiprogramm tunniplaani genereerimiseks asub lisas III ja tunniplaan JSON formaadis ning käsitsi loodud tunniplaani fail lisas IV.

[1] https://unity3d.com/unity/system-requirements

[2] https://docs.unrealengine.com/en-us/GettingStarted/RecommendedSpecifications

# 2. Nõuded

Selles peatükis on toodud Delta õppehoone keskkonna visualiseerimise funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuded olid koostatud antud bakalaureusetöö eesmärkide põhjal.

## 2.1 Funktsionaalsed nõuded

Funktsionaalsete nõuete peatükis on toodud nõuded, millele Delta õppehoone keskkonna visualisatsioon peab vastama. Funktsionaalsed nõuded on seotud visualisatsiooni väljanägemisega ning funktsionaalsusega.

F1. Ekraani vasakul poolel näidatakse õppehoone kõigis klassides parasjagu toimuvate ainete nimetusi (tunniplaan).
F2. Tunniplaan ei tohi varjata klassis käimasolevaid tegevusi.
F3. Tunniplaan peab näitama aine nimetust eesti ja inglise keeles.
F4. Visualisatsiooni integreeritakse reaalaja ilma API. Rakendus peab sobiliku liidestuse läbi olema reaalajas teadlik Tartu linna ilmast ning seda visualiseerima.
F5. Vihma sajamisel visualiseeritakse vihmapiisku, muudetakse stseeni valguse taset, õppehoone ümber luuakse lombid.
F6. Lume sajamisel visualiseeritakse lumehelbeid, muudetakse stseeni valguse taset, õppehoone ümber luuakse lumehanged.
F7. Äikese puhul muudetakse valgustase nii, et see näeks välja nagu tõeline äike.
F8. Pilvisuse korral vähendatakse valgustaset.
F9. Visualiseeritakse stseeni valguse suund järgi päikese liikumist taevas.

## 2.2 Mittefunktsionaalsed nõuded

Peatükis on toodud mittefunktsionaalsed nõuded Delta õppehoone keskkonna visualiseerimisele.

MF1. Ilma andmeid päritakse ja uuendatakse iga viie minuti järel.
MF2. Klassis toimuva aine nimetust uuendatakse iga kord kui aine vahetub.
MF3. Cumulocity sündmuseid kuulatakse reaalajas.
MF4. Klassides toimuvate ainete nimetustega silt peab olema vaatajale hästi loetav.
MF5. Tunniplaani silt ei tohi liigselt tähelepanu võtta.
MF6. Tunniplaani tekst peab olema nähtav ja loetav suurelt ekraanilt 5 meetrite juures seisva vaatleja poolt.

MF7. Tunniplaani keel peab vahetuma iga 10 sekundit.

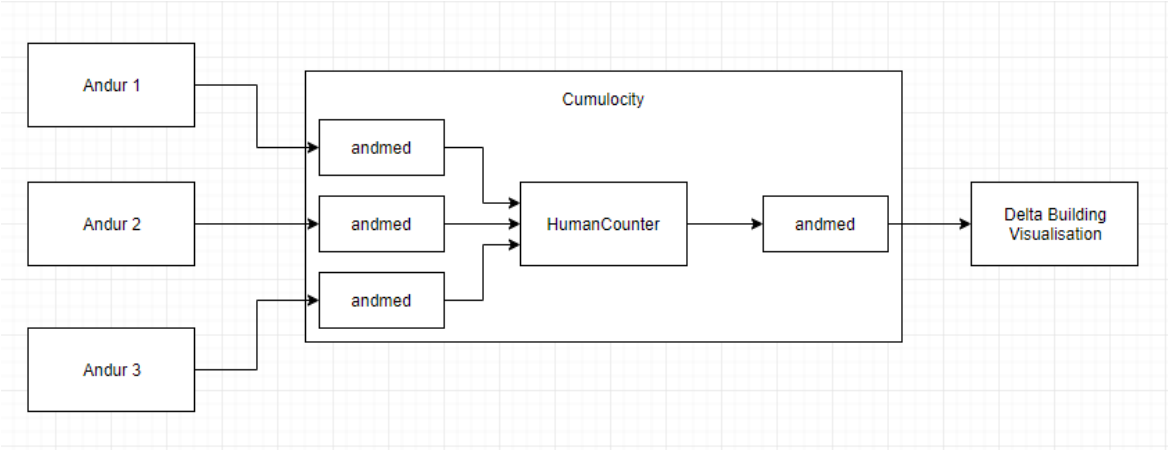
Jargmises peatükkis on kirjeldatud Cumulocity ühendamine koos Delta õppehoone keskkonna visualisatsiooniga, mille kaudu Delta õppehoone visualisatsioon kuulab väidetavaid „*HumanCounter*“ mooduli tagastusandmeid.

# 3. Cumulocity integreerimine

Cumulocity[3] on IoT[4] (*Internet of Things* ehk asjade internet) (vt. Terminid) platvorm, mis ühendab tarku seadmeid ning algoritme (antud projekti raames virtuaalsed seaded) läbi võrgu ja annab võimaluse mugavalt töötada kõikidest seadmetest tuleva informatsiooniga ühes keskkonnas. Cumulocity toetab erinevaid seadmeid, mis võivad olla konkreetse kasutusjuhuga seotud spetsiaalsed seadmed, näiteks asukoha jälgimise seadmed, või sellised arenduskomplektid nagu Raspberry Pi, Cinterion, Arduino. [5, 7]

On võimalik programmeerida endale oma programm, mis hakkab suhtlema erinevate seadmetega (nt. anduritega) läbi Cumulocity API. Cumulocity veebilehel on pakutud liidestuse teegid järgnevate keelte jaoks: Java, JavaME, C/C++, JavaScript, Rest, MQTT. Töö käigus arendati C# programmeerimiskeeles liidestus Cumulocity HTTP REST-liidestusega sündmuste kuulamise jaoks. Valik REST liidestuse kasuks tehti kasutamise lihtsuse põhjal käesoleva töö autori jaoks. Alternatiiviks HTTP REST liidestusele võimaldab Cumulocity kasutada ka MQTT protokolli. Joonisel 1 on skemaatiliselt näidatud Cumulocity platvormi töö Delta õppehoone visualiseerimisel.

## 3.1 Integratsioon




*Joonis 1: Cumulocity platvormi töö Delta õppehoone visualiseerimisel.*

Cumulocity'ga tööd alustades ühendatakse andureid, mis on paigaldatud Delta õppehoones. Antud töö tegemise ajal õppehoone on veel ehitamisel ja pole reaalseid andureid. Selle tõttu simuleeritakse väidetavaid „*HumanCounter*" mooduli tagastusandmeid. „*HumanCounter*"

[3] https://cumulocity.com/guides/concepts/introduction/

[4] https://en.wikipedia.org/wiki/Internet_of_things

inimesteloenduri algoritm tegutseb kui „virtuaalne andur“, mis võtab andurite andmeid, arvutab nendest ühe inimeste arvu konkreetses õppehoone ruumis seejärel saadab Cumulocity platvormile uue sündmuse kui arv ruumis muutub.

Selleks, et ühendada „*HumanCounter“* moodul Cumulocity platvormiga loodi C# programmeerimiskeeles Unity programm, mis saadab päringu kliendilt serverile kasutades Unity mängumootori *UnityWebRequest* klassi.

HTTP REST päringu näide uue anduri loomiseks:

Päis:

```
1. POST /inventory/managedObjects HTTP/1.1
2. Content-Type: application/vnd.com.nsn.cumulocity.managedObject+json;
   charset=UTF-8; ver=0.9
3. Accept: application/vnd.com.nsn.cumulocity.managedObject+json;
   charset=UTF-8; ver=0.9
4. Authorization: Basic <<Base64 encoded credentials <tenant
   ID>/<username>:<password> >>
```

Keha:

```
5. {
6. "c8y_IsDevice" : {},
7. "name" : "HumanCounterDevice"
8. }
```

Antud päringu saadetakse alam-aadressile */inventory/managedObjects*. Real 1 on märgitud päringu tüüp, antud päring on POST tüüpi päring. Real 4 tuleb lisada päringusse andmed *tenant*[5] ID, kasutajanime ja salasõna kohta. Selleks, et kohe visuaalselt ei oleks isiklike andmeid näha, tuleb need kodeerida Base64 algoritmi abil. Ridadega 6 ja 7 luuakse uus moodul nimega „HumanCounterDevice“. Keha osa on kirjutatud JSON formaadis.

Kui päring on saadetud, siis Cumulocity server tagastab vastuse tulemusega, kas uus andur on loodud või mitte. Kui andur on edukalt loodud, siis saadetakse järgnev vastus:

Päis:

```
1. HTTP/1.1 201 Created
2. Content-Type: application/vnd.com.nsn.cumulocity.managedObject+json; char-
   set=UTF-8; ver=0.9
```

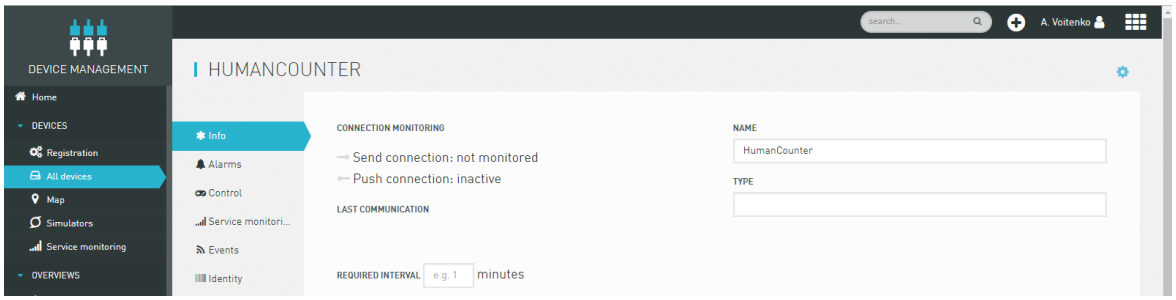
[5] https://www.cumulocity.com/guides/reference/tenants/

```
3. Authorization: Basic <<Base64 encoded credentials <tenant ID>/<user-
   name>:<password> >>
```

Keha:

```
4. {
5. "id": "2007",
6. "lastUpdated": "2018-04-01T10:58:26.279+01:00",
7. "name": "HumanCounterDevice",
8. "owner": "<username>",
9. "self": "https://<tenant-ID>.cumulocity.com/inventory/manage-
   dObjects/2007",
10. "c8y_IsDevice": {}
11. }
```

Selle päringu reast 1 selgub, et andur oli edukalt loodud, seda näitab HTTP kood[6] 201. Keha osas antakse teada, et uuel moodulil on identifikaatori number 2007 (rida 5), millal oli viimane muudatus selles anduris (rida 6), anduri nimetus (rida 7), anduri looja (rida 8), URL link (rida 9) ja saadavus hallata seadet Cumulocity liidesest[7] (rida 10). Joonisel 2 on näidatud uus moodul Cumulocity platvormis.

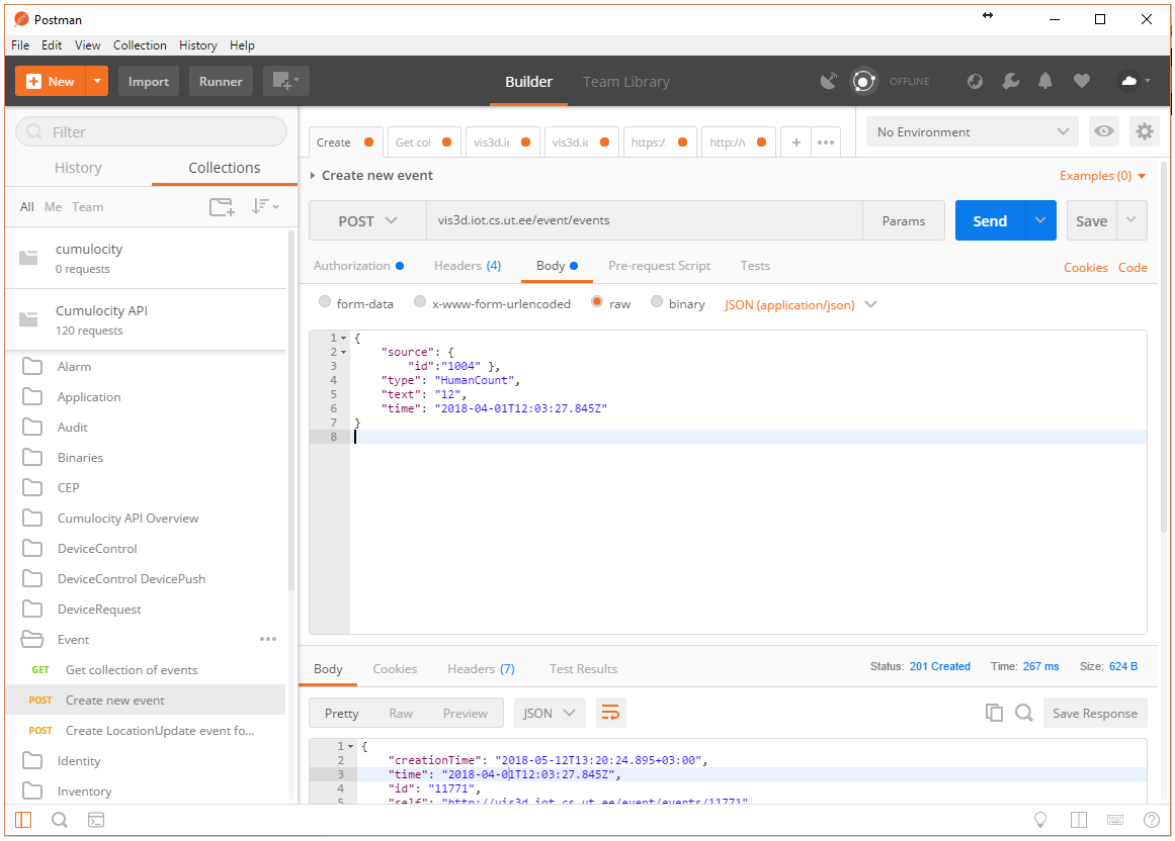



*Joonis 2: Uus moodul Cumulocity platvormis.*

Mooduli ühendamine Cumulocity platvormiga annab võimaluse töötada uue mooduliga, võtta sellest mõõdetud andmeid ja kontrollida seadme tööd. Inimesteloenduri virtuaalanduri JSON formaadis andmete Cumulocity platvormile edastamiseks on kasutatud Postman HTTP päringute koostamise ja saatmise tarkvara. Selle abil saab luua uusi sündmusi või vastupidi väljastada kõikide sündmuste listi iga Cumulocity'ga ühendatud andurilt. Joonisel 3 on näidatud uue sündmuse loomine kasutades Postman tarkvara.

[6] http://www.restapitutorial.com/httpstatuscodes.html

[7] https://www.cumulocity.com/guides/reference/device-management/



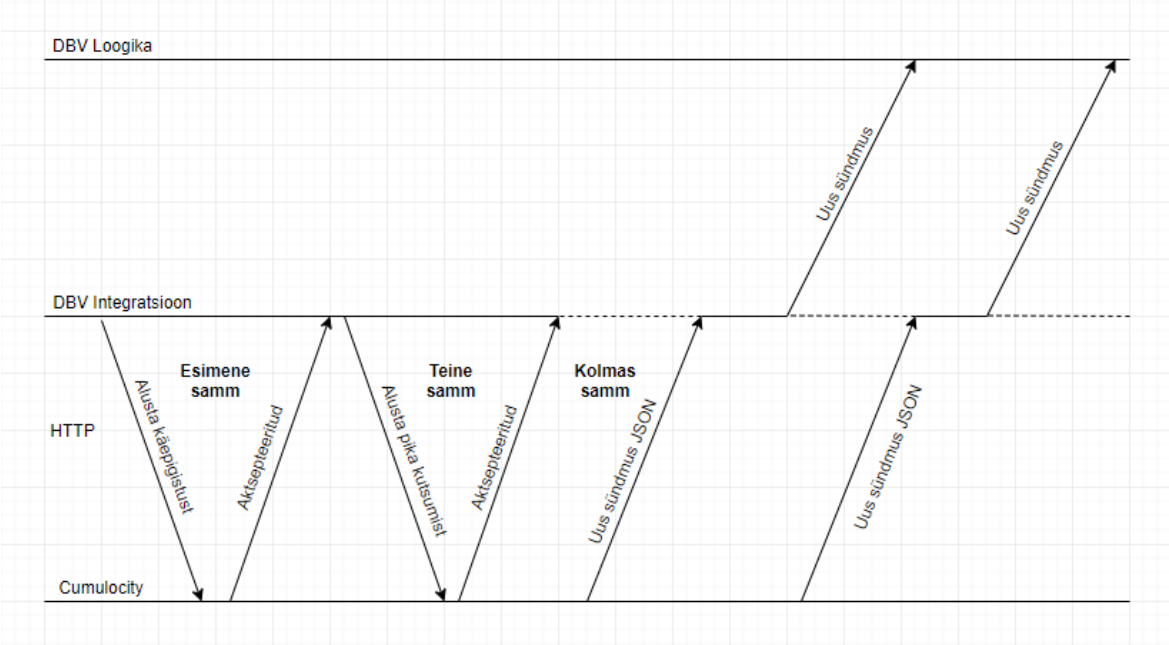

*Joonis 3: Postman sündmuse loomise vaade.*

JSON lause näide:

```
1. {
2. "source": {
3. "id":"2007"
4. },
5. "type": "HumanCount",
6. "value": "12",
7. "time": "2018-04-01T12:03:27.845Z"
8. }
```

Antud lause simuleerib uue sündmuse, mis on saadetud ruumi (*source.id*) number 2007 (rida 3) kohta. Võtmesõna *type* väärtus *HumanCount* (rida 5) annab teada, millist tüüpi sündmusega on tegu. Võtmesõna *value* väärtus tagastab inimeste arvu klassiruumis (rida 6). Sündmus luuakse iga kord, kui inimeste arv ruumis suureneb või väheneb. Võtmesõna *time* väärtus (rida 7) näitab millal sündmus oli loodud.

Andurite sündmuste kuulamiseks, luuakse C# programmi kuular (ing. k. *listener*). Cumulocity sündmuste kuulari töö printsiip on joonisel 4.




*Joonis 4: Cumulociy sündmuste kuular.*

Kõigepealt loodi Cumulocity süsteemis „sündmuste töötlemise reegel“ (ing. k. *Event Processing Rule*), mis filtreerib kõiki sündmusi ja tagastab ainult vajalike. Näide Cumulocity Event Language keeles kirjutatud reeglist:

```
select *
from EventCreated e
where e.event.type = "HumanCount";
```

Reegel tagastab kõiki sündmusi, mille tüübiks on „*HumanCount*“ („*type*“: "*HumanCount*"). Cumulocity Event Launguage (CEL[8]) on päringu keel, mida kasutatakse Cumulocity platvormis. CEL on süntaktiliselt sarnane SQL-iga, nende erinevuseks on CEL-i pidev uute sündmuste tagastamine.

Seejärel on programmeeritud kuular, mille loomisel kasutatakse REST liidestuse arhitektuuri stiili. Kuulari loomisel on kolm sammu (vt. joonis 4), mis järgivad Bayeux protokolli[9].

Esiteks saadakse POST päring serverile, mis annab teada, et luuakse käepigistust (ing. k. *handshake*):

Päis:

```
1. POST /cep/notifications HTTP/1.1
2. Content-Type: application/json
```

[8] https://cumulocity.com/guides/concepts/realtime/

[9] https://docs.cometd.org/current/reference/#_bayeux

Keha:

```
3. [ {
4. "id": "1",
5. "supportedConnectionTypes": ["long-polling"],
6. "channel": "/meta/handshake",
7. "version": "1.0"
8. } ]
```

Saadetud alam-aadressile */cep/notifications* päringu tüübiks on POST (rida 1), mille sees antakse serverile teada, et luuakse uus käepigistus päringu järjekorranumbriga id: 1 (rida 4). Käepigistus on pika kutsumisega (ing. k. *long-polling*) tüübiga (rida 5), mis tähendab rakenduse ootama jäämist kuni serverist tuleb uusi sõnumeid. Päringu kanaliks *channel* on */meta/handshake* (rida 6) ja kliendipoolseks Bayeux-i protokolli versiooniks on *1.0* (rida 7).

Eduka esimese päringu korral vastab server:

Päis:

```
1. HTTP/1.1 200 OK
```

Keha:

```
2. [ {
3. "id": "1",
4. "supportedConnectionTypes": ["websocket","long-polling"],
5. "channel": "/meta/handshake",
6. "version": "1.0",
7. "clientId": "71fjkmy0495rxrkfcmp0mhcev1",
8. "minimumVersion": "1.0",
9. "successful": true
10.}]
```

Cumulocity server vastab HTTP staatus koodiga 200 (rida 1), mis tähendab eduka käepigistuse loomist. Loodi uus käepigistus päringu järjekorranumbriga id: 1 (rida 3), selle tüübiks on *long-polling* (punkt 4), kanaliks *channel* on */meta/handshake* (rida 5) ja kliendipoolseks Bayeux-i protokolli versiooniks on *1.0* (rida 6). Edaspidiseks suhtlemiseks serveriga antakse kliendi id (rida 7), mis oli genereeritud serveris ja mida kasutatakse järgnevates päringutes. Võtmesõna *minimumVersion* väärtuseks on minimaalne vajalik serveripoolne Bayeux-i protokolli versioon (rida 8) ja pärigu edukust tähistab võtmesõna *successful* väärtus (rida 9).

Teise sammuna saadakse POST päring, mis tellib (ing. k. *subscribe*) uute sündmuste kuulamist eelnevas sammus vastuseks saadud võtmega *clientId*:

Päis:

```
1. POST /cep/notifications HTTP/1.1
2. Content-Type: application/json
```

Keha:

```
3. [ {
4. "id": "2",
5. "channel": "/meta/subscribe",
6. "subscription": "/liikumine/*",
7. "clientId": "71fjkmy0495rxrkfcmp0mhcev1"
8. }]
```

Päringu alam-aadressile */cep/notifications* tüübiks on POST (rida 1) järjekorranumbriga id: 2 (rida 4), mille kanaliks *channel* on */meta/subscribe* (rida 5). Võtmesõna *subscription* väärtuseks määratakse sündmuste töötlemise reegeli nimetus (rida 6), mis oli loodud CEL keeles. Märgitakse *clientId* väärtus (rida 7), mis tagastati serveri vastuses järjekorranumbriga id:1.

Serveri järgnev vastus:

Päis:

```
1. HTTP/1.1 200 OK
```

Keha:

```
1. [ {
2. "id": "2",
3. "channel": "/meta/subscribe",
4. "subscription": "/liikumine/*",
5. "successful": true
6. } ]
```

Server vastab HTTP koodiga 200 (rida 1), mis tähendab eduka uute sündmuste kuulamise tellimist. Päringu järjekorranumbriks on id: 2 (rida 2), kanaliks *channel* on */meta/subscribe* (rida 3), sündmuste töötlemise reegli nimetuseks on subscription väärtus (rida 4), pärigu edukust tähistab võtmesõna *successful* väärtus (rida 5).

Kolmanda ja viimase sammuna saadetakse POST tüüpi päringu, mis ühendub ja jääb pika kutsumisega (ing. k. *long-polling*) ootama uusi sündmusi andurilt:

Päis:

```
1. POST /cep/notifications HTTP/1.1
2. Content-Type: application/json
```

Keha:

```
3. [ {
4. "id": "3",
```

```
5. "connectionType": "long-polling",
6. "channel": "/meta/connect",
7. "clientId": "71fjkmy0495rxrkfcmp0mhcev1"
8. } ]
```

Seegi POST tüüpi päring saadeti alam-aadressile `/cep/notifications` (rida 1) järjekorranumbriga id: 3 (rida 4), mille ühendamistüübiks *long-polling* (rida 5) on ja kanaliks *channel* on */meta/subscribe* (rida 6). Märgitakse *clientId* väärtuse (rida 7).

Uue sündmuse saabumisel Cumulocity platvormile saadetakse see Dela õppehoone visualisatsiooni programmile. Näide saabuvast sündmusest:

```
1. [{
2.    "id": "20898490",
3.    "data": {
4.       "creationTime": "2018-04-01T15:41:55.015+03:00",
5.       "time": "2018-03-26T12:03:27.845Z",
6.       "id": "10447",
7.       "self": "http://vis3d.iot.cs.ut.ee/event/events/10447",
8.       "source": {
9.          "id": "2007",
10.         "name": "HumanCounterDevice",
11.         "self": "http://vis3d.iot.cs.ut.ee/inventory/managdObjects/10225"
12.      },
13.      "text": "12",
14.      "type": "HumanCount"
15.      },
16.   "channel": "/Liikumine/statement_1"
17.},
18.{
19.   "id": "3",
20.   "successful": true,
21.   "channel": "/meta/connect"
22.}]
```

Sündmuse vastuse reas 2 on märgitud sündmuse id, ridadel 3-15 on informatsioon sündmuse kohta, Delta õppehoone visualiseerimiseks võetakse ruumi number (*source.id*) ja sündmuse *text* väärtuse (inimeste arv). Saadud sündmuses on märgitud sündmuse ja päringu kanalid *channel* (read 16 ja 21 vastavalt), päringu järjekorranumber 3 (rida 19), pärigu edukust tähistab võtmesõna *successful* väärtus (rida 20).

Sündmuse saabumisel JSON vormingus töödeldakse seda Unity mängumootoris paigaldatud klassiteegiga (*JsonUtility*[10]), mis teeb tööd JSON lausetega kiiremaks ja mugavaks. Lauset töödeldakse ja sellest võetakse varem mainitud informatsioon (ruumi number, inimeste arv)

Pärast sündmuse saamist Unity rakenduses ja selle töötlemist, edastatakse inimeste arvud ruumides A. Nikolajevi poolt tehtud projekti ossa, kus visualiseeritakse tudengite liikumist õppehoones ja selle ruumides.

## 3.2 Tulemused

Cumulocity sündmuste kuulamise programm hakkab töötama Delta õppehoone visualisatsiooni käivitamisel ja lõpetab töö selle väljalülitamisel. Katsetati olukorda, kus interneti ühendus katkeb, sellel olukorral programm ootab uut ühendust. Ühenduse tagasitulekul jätkatakse tööd ilma vigadeta. Cumulocity integreerimise peatükis kirjeldatud töö käigus on tehtud rakendus, mis kuulab sündmusi reaalajas, mis vastab mittefunktsionaalsele nõudele MF3.

[10] https://docs.unity3d.com/ScriptReference/JsonUtility.html

# 4. Tunniplaani integreerimine ja visualiseerimine

Selleks, et õppejõududel, tudengitel ja teistel Delta õppehoone külastajatel oleks lihtsam aru saada, mis toimub igas konkreetses klassiruumis, luuakse tunniplaan, kus on igas klassiruumis parasjagu toimuva aine nimetus. Delta õppehoone visualiseerimises reaalaja tunniplaani näitamiseks tuleb programm liidestada Tartu Ülikooli Õppeinfosüsteemiga (edasi ÕIS). Töö kirjutamise hetkel ei ole Delta maja valmis ning ka ÕIS-i uus versioon 2.0 alles arenduses.

Tartu Ülikooli õppeinfosüsteemi arendajaga (Gert Post) suhtlemisel selgus, et uue ÕIS-i versiooni loomisel ÕIS 2.0 arendajad loovad sellise API, mis võimaldaks arendajatel saada kätte tunniplaani iga ruumi kohta otse õppeinfosüsteemist.

Kuna töö tegemisel ei olnud võimalik rakendust liidestada päris süsteemiga, siis loodi simuleeritud tunniplaan (vt lisa V) õppehoone iga ruumi ja nädalapäeva kohta. Tunniplaani töödeldakse ja visualiseeritakse. Visuaalselt uuendatakse seda iga kord, kui mõnes klassis algab uus aine.

## 4.1 Integratsioon

Seoses ÕIS 2.0 liidestuse puudusega töö tegemisel, luuakse JSON fail, mille sees on genereeritud tunniplaan. Fail sisu luuakse käsitsi failis mis on tööga kaasas lisas IV, aga ainete nimetused ja registreeritud tudengite arvud muudetakse Python programmiga, mis on antud tööga kaasas lisas IV.

JSON tunniplaani sisu enne Python programmi töötlemist:

```
{
    "Monday":{
        "Rooms":[
            {
                "Id":"1004",
                "Subject":[
                    {
                        "Start":"08:15 UTC+2",
                        "Name":"Aine",
                        "Students":"--",
                        "Teacher":"Name"
                    },
                    {
                        "Start":"10:15 UTC+2",
                        "Name":"Aine",
```

```
                    "Students":"--",
                    "Teacher":"Name"
                },
                ...
            ]
        },
        ...
    ]
},
...
}
```

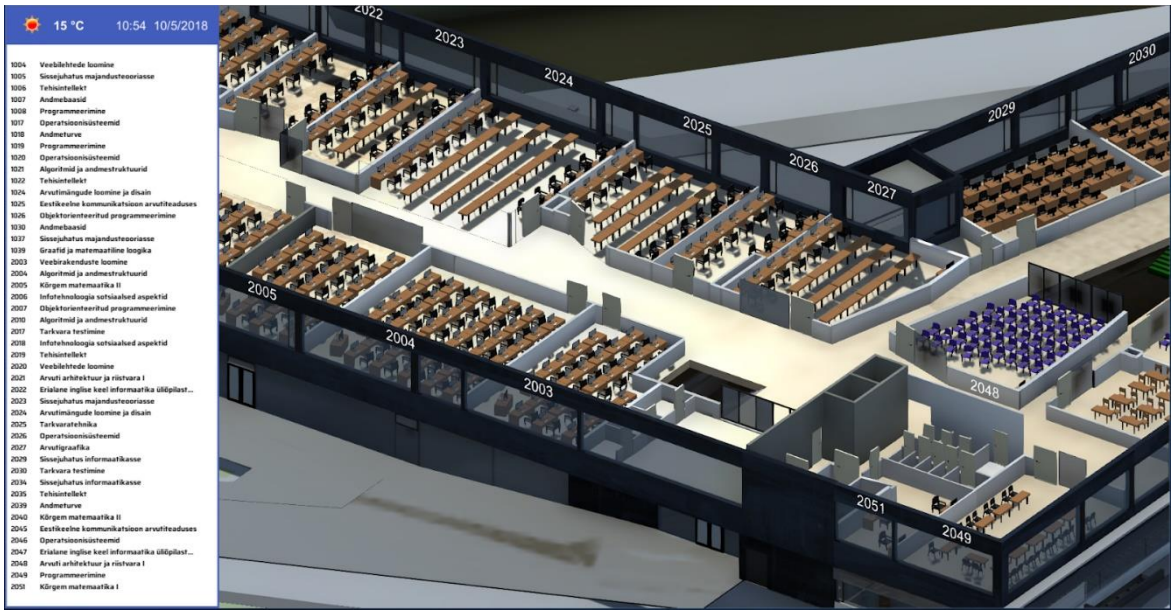
Python programm paneb ainetele juhuslikud nimetused etteantud massiivist ning lisab inglise keelsed nimetuse ja aine tüübi. Aine tüübiks on praktikum (*practical session*), seminar (*seminar*), loeng (*lecture*), kontrolltöö (*test*), eksam (*exam*). Lisaks programm muudab tudengite arvu „--“-st juhusliku numbri peale vahemikust [7, 50] ning lisab õppejõudude suvalised ees- ja perekonnanimed. Töödeldud JSON tunniplaani sisu on järgnev:

```
{
    "Monday":{
        "Rooms":[
            {
                "Id":"1004",
                "Subject":[
                    {
                        "Start":"08:15 UTC+2",
                        "Name":" Arvutigraafika",
                        "Name_Eng":" Computer Graphics",
                        "Type":" lecture",
                        "Students":35,
                        "Teacher":" Jüri Pärn"
                    },
                    {
                        "Start":"10:15 UTC+2",
                        "Name":" Andmeturve",
                        "Name_Eng":"Computer Security",
                        "Type":"seminar",
                        "Students":24,
                        "Teacher":" Jüri Pärn"
                    },
                    ...
                ]
            },
            "Id":"1005",
            ...
```

```
28.        ]
29.    },
30.    "Tuesday":{
31.    ...
32.}
```

Ülal toodud tunniplaan on viie tööpäeva kohta (esmaspäev kuni reede, read 2 ja 30. Ülejäänud päevade kohta vt. lisa IV). Iga päev on jaotatud ruumideks (rida 3) ning igas ruumis on seal toimuvate ainete list (read 5-6). Listis on märgitud aine alguse kellaaeg (read 8, 16), nimetused eesti ja inglise keeles (read 9 - 10, 17 - 18), aine tüüp (read 11, 19), õppetööle (loeng, praktikum, seminar, kontrolltöö, eksam) registreerunud tudengite arv (read 12, 20) ja õppejõu nimi (read 13, 21).

Pärast tunniplaani loomist JSON formaadis, töödeldakse andmeid Unity rakenduses. Tunniplaani visualiseerimiseks jagatakse ekraan kaheks osaks. Üks osa, mille mõõdud on 350px × (ekraani pikkus), on tunniplaani visualisatsiooniks. Teine osa, mille mõõdud on (ekraani laius - 350px) × (ekraani pikkus), kasutatakse õppehoone 3D visualisatsiooni jaoks. Joonisel 5 on näidatud tunniplaan Delta õppehoone visualisatsioonil.




*Joonis 5: Tunniplaani ja ruumide numbrite nähtavus visualisatsioonil.*

Tunniplaani visualiseerimiseks kasutatakse *UI.Canvas* elementi, mille peal asub paneel *UI.Panel*. Värvid on üks efektiivsematest teguritest, millega saab mõjutada inimeste emotsioone [8]. Selleks, et värv oleks neutraalne, ei tekitaks ebavajalike emotsioone ja ei võtaks liigselt tähelepanu, on paneeli tausta värviks valitud valge värv (hex #FAFAFA).

Paneelile paigaldatakse tunniplaani tekst, mis on moodustatud Unity kahest Unity UI teksti komponendist. Üht *UI.Text* komponendi kasutatakse klassiruumide numbrite visualiseerimise jaoks, teist aine nimetuse visualiseerimise jaoks. Selleks, et tähed oleksid kaugelt loetavad, kasutatakse musta värvi (hex #333333) ja erinevaid fonte. Ruumi numbrite jaoks kasutatakse *SairaSemiCondensed-SemiBold* font ja ainete nimetuseks *Siera-SemiBold* font. Selline valik on põhjendatud nende fontide kaugelt loetavusest, sellise järelduseni tuli autor pärast erinevate fontide kaugelt loetavuse katsetamist. Tähtede automaatse suuruse leidmiseks kasutatakse *Unity.Text* funktsiooni *Best Fit*, mis leiab tekstivälja suurusele vastava tähtede suurust. Minimaalseks tähtede suuruseks on 5 pt ja maksimaalseks 20 pt ning reavaheks 1. Tunniplaani visualisatsioonil olevaid fonte ja tähtede suuruste näited on näidatud joonisel 6.

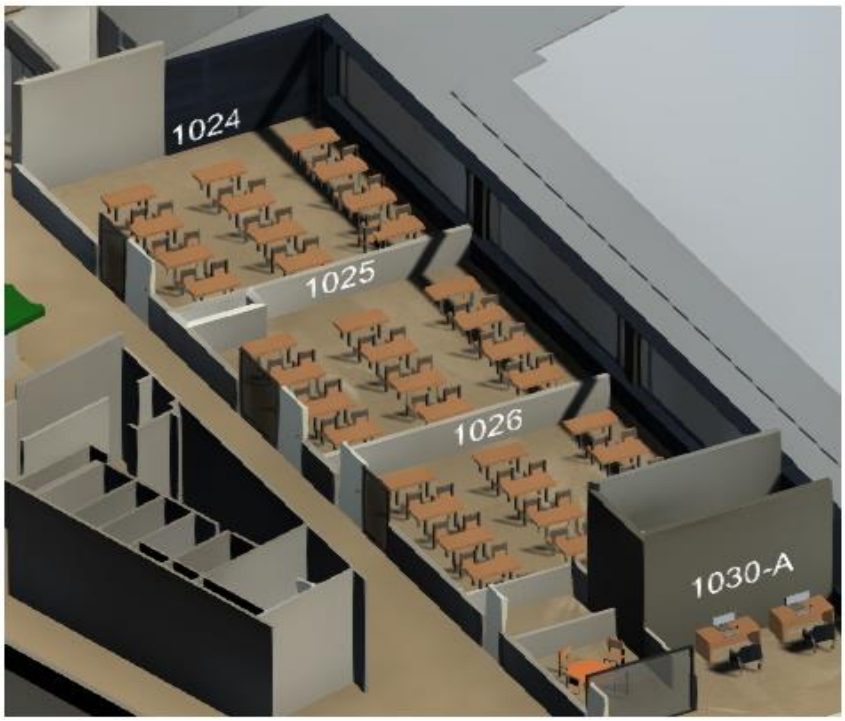

*Joonis 6: tunniplaani visualisatsiooni fonti näide.*

Tunniplaani tekstiks kasutatakse andmeid varem loodud JSON failist. Ainete nimetuste õigesti kuvamiseks on loodud C# skript, mis võtab hetke aja, võrdleb seda JSON tunniplaaniga ja visualiseerib õige aine nimetuse (kontrollitakse nädalapäeva, kellaaega, ruumi numbrit). Kuupäevi programm ei kontrolli, sest reeglina tunniplaan kordub iga nädal, eranditks on eksamipäevad. Eksamipäevi antud programm ei arvesta selle pärast, et tihti neid päevi pannakse semestri käigul, sellel ajal ÕIS tunniplaan on juba koostatud.

Iga 10 sekundi järel tunniplaani uuendatakse ja vahetakse keelt. Tunniplaani uuendamine toimub selleks, et aine nimetused muutuksid õigeaegselt aine vahetuse korral. Tunniplaani keelt omakorda vahetatakse, et eesti keelt mitte valdavad välistudengid saaks ainete nimetustest aru. Kui klass on vaba ja seal pole tunde, siis tunniplaanil seda klassiruumi ei näidata.

Tunniplaan on visualiseeritud nii, et planeeritud suurel ekraanil, mis on analoogne Tartu Ülikooli raamatukogus paigaldatud ekraaniga (selle suuruseks on 55 tolli), kõikide klassiruumide (neid on 46) ainete nimetused on näidatud korraga. Üks aine nimetus on ühes reas, kui nimetuse pikkus on suurem kui 40 tähte ja ei mahu reale, siis ekraanile kuvatakse aine nimetuse esimesed 40 tähti ja lisatakse „…“, tingimusel et ei jääks üksikut tähte või tühikut enne „…“. Kui jääb tühik koos üksik tähega, siis kuvatakse ekraanile esimesed 43 tähe, kui jääb tühik, siis 44.

Klassiruumide numbrite identifitseerimiseks 3D mudeli visualisatsioonil on seintele pandud tekstilised märgendid, mis tähistavad ruumide numbreid, need on nähtavad joonisel 7. Seintele numbrite peale panemiseks kasutatakse *UI.Canvas* komponenti mille renderdusrežiimiks (*Render Mode*) on maailma ruum (*World Space),* seda tehakse numbrite kinnitamiseks 3D mudelile. *UI.Canvas* renderdusrežiimiks on sellised seaded, mida saab kasutada ekraanirummi (*screen space*) või maailmaruumi (*World Space*) renderdamiseks. Nüüd kaamerate muutmisel ja liikumisel jäävad kabinettide numbrid samale asukohale maailma ruumis.




*Joonis 7: Ruuminumbrite märgendid.*

Lisaks on tunniplaani ülemises osas kuvatud reaalaja kellaaeg ja kuupäev selleks, et visualisatsiooni vaatajatel oleks informatsioon kellaajast ja kuupäevast ilma kella ja kalendri vaatamiseta.

## 4.2 Tulemused

Tunniplaani integreerimisel visualiseeriti tunniplaan, mis näitab iga õppehoone klassiruumides hetkel toimuvate aine nimetusi. Tartu Ülikooli raamatukogu ekraanidel katsetati klassiruumide numbrite ja nimetuste nähtavust suurel ekraanil 5 meetri kauguselt. Tulemus vastas vastavale mittefunktsionaalsele nõudele MF6. Tunniplaan ei kata 3D Delta õppehoone visualisatsiooni ja ei võta liigselt tähelepanu, mis vastab mittefunktsionaalsele nõudele MF5. Tunniplaan vahetab aine nimetuste keelt iga 10 sekundi järel, mis vastab funktsionaalsele nõudele F3 ja mittefunktsionaalsele nõudele MF7. Sai rahuldatud ka funktsionaalsed nõuded F1 ja F2 ja mittefunktsionaalne nõude MF2.

Tunniplaani pikaajalisel katsetamisel tuli välja üks töö puudus. Tunniplaani uuendatakse iga 10 sekundi järel, aga kui tunniplaan just uuenes ja vähem kui 10 sekundi jooksul aine ruumis muutus, siis tunniplaan näitab veel 10 sekundi lõpuni vana aine nimetuse ja järgmisel uuendusel muudab seda õige nimetuse peale. Päris olukorral ei tohiks olla see probleem, sest maksimaalselt 10 sekundi pärast tunniplaan muutub korrektseks, selle aja jooksul .

Tunniplaani ning klassiruumide numbrite nähtavuse katsetamiseks Tartu Ülikooli raamatukogu ekraanidel olid kutsutud viis inimest kes andsid oma arvamust tehtud visualisatsioonist. Testeerijatele oli pandud käima Delta õppehoone keskkonna visualisatsioon, millel testeeriti tunniplaani nähtavust erinevatest kaugustest. Arutelust tuli välja, et tunniplaanil olevad aine nimetused ja klassiruumide numbrid on halvasti loetavad. Selle katse tulemuse põhjal käesoleva töö autor muutis fonti ning tegi tunniplaani ekraani osa laiemaks.

# 5. Keskkonna visualiseerimine

Keskkonna visualiseerimiseks on visualisatsioon liidestatud OpenWeatherMap[11] ilmateenuse rakendusliidesega. OpenWeatherMap on VANE Geospatial Data Science platvormil põhinev ilmastiku teenus, mis kogub, töötleb, ja väljastab ilma informatsiooni HTTP REST rakendusliideste abil. OpenWeatherMap API annab kasutajatele võimaluse saada andmeid JSON või XML vormingus.

Delta õppehoone 3D visualisatsioonis kasutatakse töödeldud OpenWeatherMap API reaalaja ilma andmeid Delta õppehoone keskkonna visualiseerimiseks. Keskkonna visualiseerimine tähendab ilmasiku ja öö/päeva tsükli integreerimist õppehoone 3D mudeli visualisatsioonis.

Öö/päeva tsükli integreerimine tähendab virtuaalse suundvalgusallika suuna muutmist vastavalt reaalse päikese asukohale Delta õppehoone suhtes. Suunavalgus[12] (ing. k. *Directional Light*) on valgusallikas, mida kasutatakse arvutigraafikas valgusefektide loomiseks, näiteks päikesevalguse jaoks. Suunavalgus valgustab objekte lõpmata kaugelt ja seda saab paigaldada Unity stseenis mistahes koordinaatidele. Kõik objektid, mida valgustab suunavalgus, on valgustatud ühest suunast. Lisaks kellaajast sõltuvale päikesevalguse suunale tuleb arvestada ka aastaringsete erinevustega. Nimelt talvel on päike madalamal ja ööd pikemad ning suvel vastupidi.

## 5.1 OpenWeatherMap API liidestus

Informatsiooni reaalaja ilma kohta saamiseks saadetakse päring OpenWeatherMap serverile, mis on täiendatud muutujatega. Näide sobivast URL lingist: *http://api.openweathermap.org/data/2.5/weather?q=Tartu&mode=json&units=metric&APPID=7cc077cc0e9b344bd8457f966ced0a4b*. URL muutujate selgitus on toodud tabelis 2.

[11] https://openweathermap.org/technology

[12] https://docs.unity3d.com/Manual/Lighting.html

*Tabel 2: OpenWeatherMap lingi muutujate selgitused.*

| **Lingi muutujad** | **q** | **mode** | **units** | **APPID** |
|---|---|---|---|---|
| **Muutujate tähendused** | linna nimetus | väljastava informatsiooni vorming | temperatuuriskaala ühik | API võti |

Järgmisena saadetakse link serverile Unity skripti abil, selleks kasutatakse Unity mängu-mootoris oleva klassiteeki. On olemas kaks võimalust ühendada serveriga: C# klassiteek *System.Net* ja Unity klassiteek *UnityWebRequest*. Sarnaselt Cumulocity integratsiooniga, on ka siin kasutatud Unity mängumootori *UnityWebRequest* klass. Pärigu vastuseks saadud andmed koosnevad reaalaja ilma informatsioonist teatud linna asukohal. Näide saadud andmetest JSON formaadis on antud tööga kaasas lisas VI.

Vajalike andmete eraldamiseks töödeldakse JSON laused, kust võetakse muutujaid mis on toodud ja selgitatud tabelis 3.

*Tabel 3: muutujate selgitused.*

| **Parameeter (ühik)** | **Selgitus** |
|---|---|
| Temperatuur (°C) | Lumehangede kasvamiseks/sulemiseks |
| Ilma kirjeldus | Ilma tuvastamiseks, sademete visualiseerimisels |
| Pilvisus (%) | Päikese heleduse muutmiseks |
| Ilma ikoon (*weather.icon*) | Reaalaja ilma ikooni ekraanile panemiseks |

Järgmise sammuna visualiseeritakse saadud andmete põhjal Delta õppehoone keskkonda.

## 5.2 Ilmastiku visualiseerimine

Ilmastiku efektide visualiseerimiseks kasutatakse Unity osakestesüsteeme[13] (ing. k. *particle system*), mille abil visualiseeritakse lume ja vihma sademeid.

Osakeste süsteem on tehnika, mida kasutatakse arvutigraafikas objektide esitamise viisina, millel pole selgeid geomeetrilisi piire. Osakestesüsteemidega saab visualiseerida pilvi, udu,

[13] https://en.wikipedia.org/wiki/Particle_system

plahvatusi, suitsu, lund, vihma jm. Neid võib kasutada nii kahemõõtmelises kui ka kolmemõõtmelises graafikas.

Osakesed (ing. k. *particles*) on väiksed, lihtsad objektid (tüüpiliselt pildid) millele antakse konkreetne tegevus (liikumine, värvi vahetamine, suuruse ja kiiruse muutmine, pöörlemine, allumine gravitatsioonile jne.). Reaalajas ilma efektide käivitamiseks on loodud skript, mille ülesandeks on ilma jälgimine ja õigete efektide käima panemine. Kõik ilma variatsioonid on toodud tabelis 4.

*Tabel 4: Ilma variatsioonid.*

| **Nimetus** | **Osakestesüsteemi efektid** | **Suundvalgusallika heledus** | **Muu** |
|---|---|---|---|
| Päikesepaiste | - | maksimaalne | kui on olemas lombid/lumehanged ja õhutemperatuur > 0°C, siis need järk-järgult kaovad |
| Pilvisus | - | vähendatud | |
| Vihmasadu | vihmasaju osakestesüsteem | vähendatud | tekivad lombid,<br>välgu efekt kui on äike |
| Lumesadu | lumesaju osakestesüsteem | vähendatud | tekivad lumehanged |

Järgmistes peatükkides on kirjeldatud ilma variatsioonide loomise protsessid.

### 5.2.1 Päikesepaiste

Päikese visualiseerimiseks kasutatakse suundvalgusallikat, heledust reguleeritakse C# skripti abil, mis vähendab/suurendab heledust vastavalt OpenWeatherMap teenusest saadud andmetele. Päikesetõusu ning -loojangu ajal päikse värvus muutub punaseks või oranžiks [9]. Seega suundvalgusallikas muudab oma värvust iga hommik, keskpäev ja õhtu, nii päikeseloojangu ja -tõusu ajal päike on kollane punase tooniga (hex #FFA99FF) ja keskpäeval helekollane (hex #FFF4D6FF). Päikese värvi muutmine toimub iga hommik kell 9:00, millal päike saab kollaseks 30 minuti jooksul ning iga õhtu kell 18:00, millal päike muutub kollaseks punaste toonidega. Valitud värvid kõige rohkem meenutavad päikese valgust

vastavalt kellaajale, sellise järelduseni jõudis töö autor pärast suunavalgus allika värvide katsetamist. Päikese sujuvaks värvi muutmiseks kasutatakse *Lerp* funktsioon mis tähendab lineaarset interpoleerimist.

### 5.2.2 Pilvisus

Pilvede efekti Delta õppehoone keskkonna visualiseerimisel saavutatakse suunavalgusallika heleduse vähendamisega. Pilvisuse taset saadakse teada OpenWeatherMap teenusest saadud ilmastiku andmetest (*clouds.all*). Vastavalt saadud andmetele reguleeritakse Unity suunavalgus allika heledust (*Light.Intensity*). Parameetri (*Light.Intensity*) väärtuseks pannakse 1 − (*clouds.all* ÷ 200), selle põhjuseks on visualisatsiooni liiga tumedamaks muutmine, seega maksimaalse pilvisuse korral (*clouds.all* = 100%) suunavalgusallika heleduse parameetri väärtuseks 0.5.

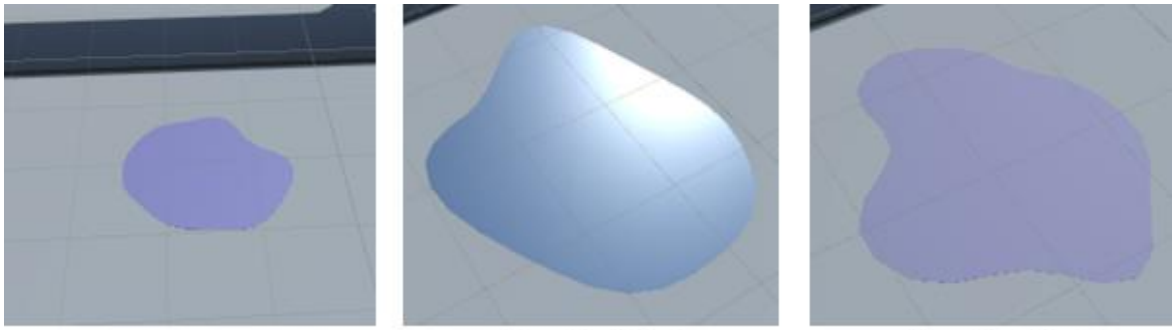
### 5.2.3 Vihmasadu

Vihma efekti loomiseks kasutatakse osakestesüsteemi, mille materjaliks on loodud vihmatilkade tekstuur (vt joonis 8). Selleks, et vihmatilgad lendaksid ülevalt alla on gravitatsiooni parameetriks (*Gravity Modifier*) märgitud 100.

*Joonis 8: vihmapiiska tekstuur.*

Vihmaga, mille kestvus ületab viis minutit, tekivad lombid, nende suurus kasvab ajaga. Lombide 3D mudelid olid loodud Blender [14]tarkvaras, mis võimaldab luua 3D objekte ning teha nende objektide animatsioonid. Lombi tekkimisest lõpliku suuruseni läheb 900 sekundid (15 minutit) ning hakkavad aurustuma kui on päikesepaisteline ilm kestusega 500 sekundid (8,3 minutit). Lombi aurustumise aja algusest kuni lõpliku kadumiseni läheb 15 minutit. Lombide tekkimise/aurumise animatsioonid on tehtud Unity Animator ja Unity Animation tööriistadega. Animatsioone käivitatakse kui C# scriptis antakse lompide objektidele triger (ing. k. *trigger*), mis vastab lompide tekkimisele (*inc* triger) või aurumisele (*dec* triger). Lompide näiteid on joonisel 9. Joonisel 10 on näidatud vihmane ilm, kus sajab vihma ja on tekkinud lombid.

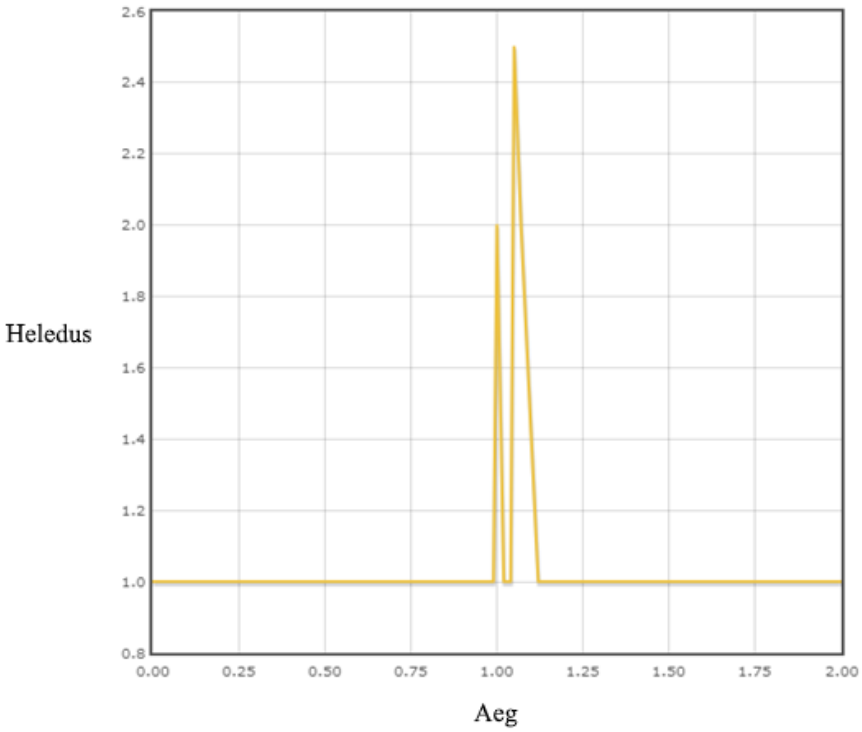
[14] https://en.wikipedia.org/wiki/Blender_(software)




*Joonis 9: Lompide näited*


*Joonis 10: Vihmane ilm visualisatsioonis.*

Äikeselise ilma korral visualiseeritakse välgu efekti iga juhusliku aja järel vahemikust [1, 100] sek. Välgu efekt on saavutatud C# skriptiga suundvalgusallika heleduse muutmisel. Äikeselise ilma oleku tähistab OpenWeatherMap andmetes read 7-10 (vt. lisa VI). Joonisel 11 on näidatud heleduse muutmine vastavalt ajale.




*Joonis 11: suunavalgus allika heleduse muutmine välgu korral.*

Vihmase ilmastiku korral vähendatakse suunavalgusallika heledust vastavalt pilvisusele (vt peatükk 5.2.2).

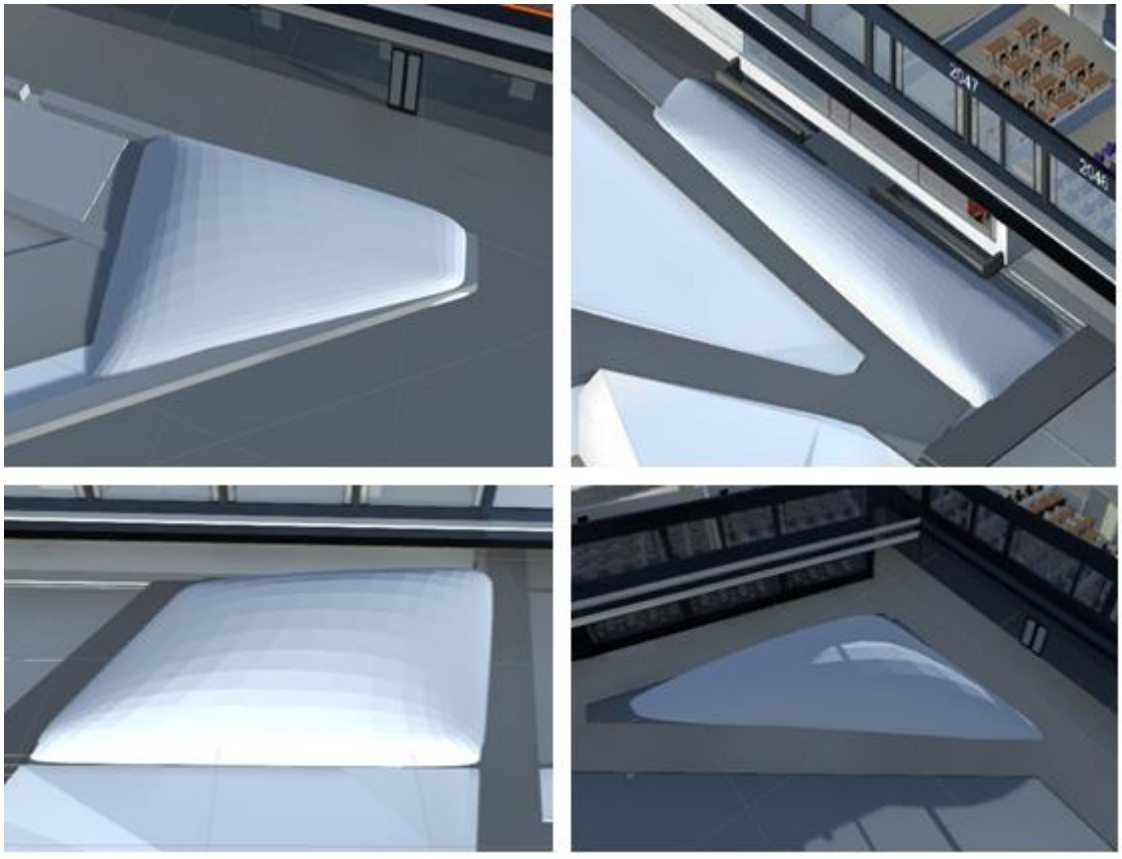
### 5.2.4 Lumesadu

Lumesaju efekti loomiseks kasutatakse osakestesüsteemi, mille materjaliks on loodud lumehelbe tekstuur (vt joonis 12). Lumehelbete realistlikuks langemiseks kasutatakse osakestesüsteemi *Velocity over Lifetime* muutuja väärtuseks kaks parameetrit: $(x, y, z) = (-1, -2, -1)$ ja $(x, y, z) = (1, -10, 1)$ mis määravad osakeste langemise suuna ajas.

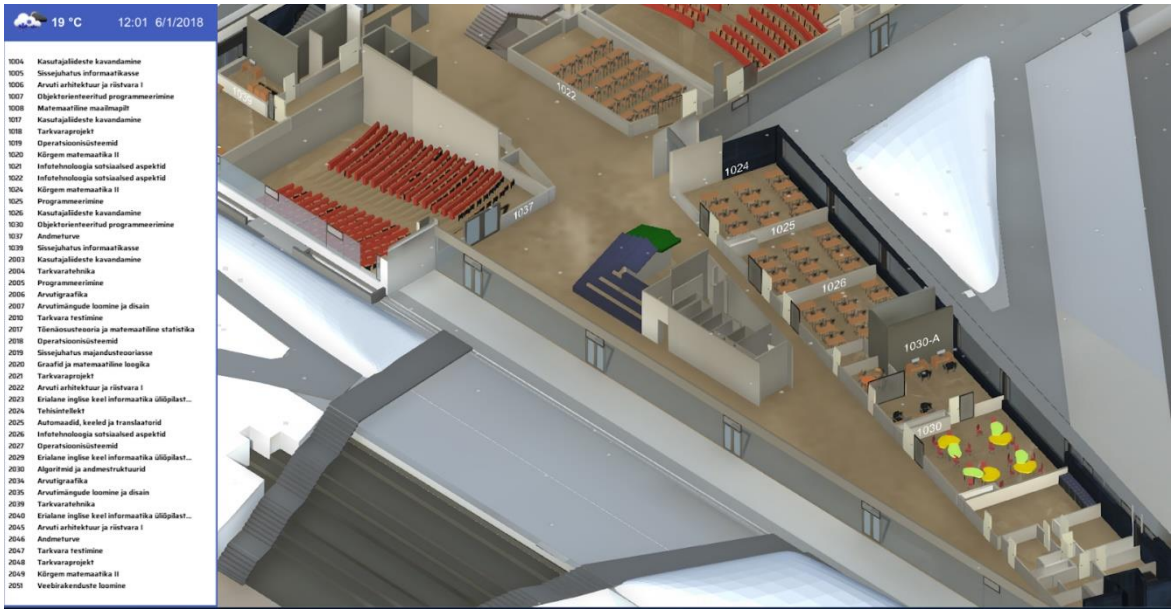
*Joonis 12: lumehelbe tekstuur.*

Lumehelbete kogust määrab parameeter *Max Particles* mida suurendatakse lumesaju algamisel ja vähendatakse selle lõppemisel. *Max Particles* väärtuseks lumesaju ajal on 6000, mis tähendab 6000 osakese tekkimist visualisatsioonil korraga, aga ekraanil on näha ainult osa sellest väärtusest, sest paljud osakesed on kaamera vaateväljast väljas.

Lumesaju ilmaga, mis kestab rohkem kui 17 minutit hakkab muutma rohu värvus (roheline → valge), et tekiks efekt, millal lumi hakkab katma murd. Rohu värvi läheb 60 minutit. Selleks kasutatakse C# oleva funktsiooni *Lerp(material a, material b, float t)*, kus materjal *a* ja materjal *b* on vastavalt muru ja lume Unity materjalid, ujukomaarv *t* on materjalide muutmise protsent. Pärast rohu värvuse muutmist ja miinus kraadide õhu temperatuuri korral viie tunni jooksul hakkavad tekkima lumehanged, mis olid loodud sarnaselt lompidega Blender tarkvaras. Lumehangede näiteid on joonisel 13. Joonisel 14 on näidatud visualisatsiooni talvine ilm, kus sajab lund, rohi on katud lumega, on olemas lumehanged.



*Joonis 13: Lumehangede näited.*




*Joonis 14: Talvise ilma visualisatsioon.*

Lumehanged hakkavad sulama 5 tunni jooksul õhu temperatuuri pluss kraadide tingimusel. Pärast lumehangede sulamist hakkab muutma muru värv roheliseks 180 minuti jooksul.

## 5.3 Öö/päeva visualiseerimine

Öö/päeva visualiseerimise jaoks tuleb muuta päikesevalgust simuleeriva suundvalgusallika suunda. Selleks, et päikese valguse suund vastaks reaalmaailma päiksele kasutatakse olemasoleva GitHub skripti Sun.cs[15], mille autoriks on Paul Hayes. Antud skript annab valgusallikale reaalse päikese suuna vektori koordinaadid konkreetses asukohas. Delta õppehoone keskkonna visualiseerimisel on asukoht Tartu linna koordinaatidega (laiuskraad: 58° 23’ 00” N, pikkuskraad: 26° 43’ 00” E).

## 5.4 Tulemused

Ilma visualiseerimise peatükis kirjeldatud töö käigus sai tehtud ilmastiku efektid ja animatsioonid, mis vastavad funktsionaalsetele nõuetele F4, F5, F6, F7, F8, F9 ja mittefunktsionaalsele nõuele MF1. Töö raames katsetati erinevad ilmastikuolsi ja nende vahetusi (nt. vihmasadu → lumesadu), programm ei väljasta vigu ja muudab ilmastikku sujuvalt.

Lisaks sai pandud tunniplaani ülemisse ossa, kus on kellaaeg ja kuupäev, ilmastiku pilt ja õhu temperatuur Tartus. Ilmastiku pilt ja õhutemperatuur aitavad aru saada milline on

[15] https://gist.github.com/paulhayes/54a7aa2ee3cccad4d37bb65977eb19e2

väljaspool Delta õppehoonet ilm. Pilte võetakse OpenWeatherMap teenusest pakutavatest piltidest, mida saadakse kasutades *UnityWebRequest* klassi.

Joonisel 15 näidatud pildid visualiseerivad selget ilma (pilt 1), vahelduva pilvisust (pilt 2), selgimistega pilvisust (pilt 3), pilvisust (pilt 4), vihmasaju (pilt 5), hoovihma (pilt 6), äikest (pilt 7), lumesaju (pilt 8) ja udu (pilt 9).


*Joonis 15: Ilmastiku pildid OpenWeatherMap teenusest.*

Piltide ja ilmastiku andmete saamiseks kasutatakse Interneti ühendust. Selleks, et visualisatsioon ei väljastaks vigu ühenduse puudumisel oli loodud skript, mis kontrollib ühendust ning selle olemasolul uuendatakse pilt ning ilmastiku andmeid. Ühenduse puudumisel programm ootab uue Interneti ühenduse.

# 6. Kokkuvõte

Käesolev töö kirjeldab visualisatsiooni loomise protsessi, mida võetakse kasutusele Tartu Ülikooli Delta õppehoones. Töö raames valmis ühendus Cumulocity platvormi ja Delta õppehoone visualisatsiooni vahel mille kaudu edastatakse „*HumanCounter*“ väärtuse asendavaid *mock* andmeid Cumulocity platvormist õppehoone visualisatsiooni.

Töö käigul valmis *mock* tunniplaan ja potentsiaalne integratsioon ÕIS 2.0-ga. *Mock* tunniplaan oli loodud Delta õppehoone klassiruumide tunniplaani visualisatsiooniks, mis näitab igas klassiruumis hetkel toimuva aine nimetust ja klassiruumi numbrit kus aine toimub. Klassiruumide asukoha paremaks arusaamiseks olid disainitud sildid, mis tähistavad ruumide numbreid 3D visualisatsioonil.

Integreeriti ilmastiku visualiseerimist, mis põhineb OpenWeatherMap rakendusliidese andmetel. Realiseeriti päikeseloojangu ja -tõusu visualisatsiooni, mille ajal suundvalgusallika värvi toon muutub punaseks. Integreeriti pilvisust, mille korral suundvalgusallika heledus väheneb vastavalt ilma andmetele. Vihmasaju ja lumesaju visualiseerimiseks kasutati osakestesüsteemi, mille tekstuuriks valmistati vihmapiiskade ja lumehelbede 2D tekstuurid. Lompide ja lumehangede visualiseerimiseks on loodud lombipide ja lumehangede 3D mudelid. Päikese valguse suunda vastamiseks reaalmaailma päiksele kasutatakse C# skript mis annab valgusallikale reaalse päikese suuna vektori koordinaadid.

Käesoleval tööl on edasi arendamise võimalus, selleks võib integreerida näiteks sündmuste (nt. Eesti riigipühad) ja ürituste (nt. bakalaureusetööde kaitsmised, tudengipäevad jne.) visualisatsiooni, mille tulemuseks muudetakse inimeste mudelite käitumist ja õppehoone 3D mudeli väljanägemist.

Tänan Tartu Ülikooli *The Computer Graphics and Virtual Reality Lab* praktilise osa valmistamiseks kasutusse antud arvutite eest ning Tartu Ülikooli raamatukogu suurel ekraanil visualisatsiooni testeerimise eest. Tänan oma juhendajat Raimond-Hendrik Tunnel, kes aitas mind kogu lõputöö tegemise ajal ning vastas kõikidele töö raames tekkinud küsimustele.

# 7. Viidatud kirjandus


[1] Blewett R. The Importance of Data Visualization to Business Decision Making, 2011. http://www.dashboardinsight.com/Article.aspx?id=4147 (04.05.2018)

[2] Robertson G. G, Mackinlay J. D, Card S. K. Cone Trees: animated 3D visualisations of hierarchical information. https://research.tableau.com/sites/default/files/p189-robertson.pdf

[3] Nikolajev A. Delta Building Model Visualisation and Optimization. Tartu Ülikool, bakalaureusetöö 2018.

[4] Creighton R. H. Unity 3D Game Development by Example. Birmingham: Packt Publishing, 2010.

[5] Kaldvee M. Cumulocity platvorm. Tartu Ülikool, bakalaureusetöö 2017. http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=60601&year=2017 (28.04.2018)

[6] Vasishta A. Which should I learn: OpenGL or Game Engine (Unreal Engine)? 2015. https://www.quora.com/Which-should-I-learn-OpenGL-or-Game-Engine-Unreal-Engine (13.05.2018)

[7] Schreieck M, Hakes C, Wiesche M, Krcmar H, Governing Platforms in the Internet of Things. Garching: Springer-Verlag Berlin Heidelberg. 2011. https://www.excell-mobility.de/wp-content/uploads/2017/06/Submission-5-Governing-Platforms-in-the-Internet-of-Things.pdf (14.05.2018)

[8] Kurt S, Osueke K. K. The Effects of Color on the Moods of College Students, 2014. http://journals.sagepub.com/doi/abs/10.1177/2158244014525423 (28.04.2018)

[9] Corfidi S. F. The Colors of Sunset and Twilight. 2014. http://www.spc.noaa.gov/publications/corfidi/sunset/ (14.05.2018)

# Lisad

## I. Terminid

| |
|---|
| **API** (*Application Programming Interface*)<br><br>Rakendusliides, mille kaudu saab teha päringuid ühest programmist teisele. |
| **IoT** (*Internet of Things*) ehk **asjade internet**<br><br>Seotud tarkade seadmete võrk ühendatud koos Interneti kaudu. |
| **REST**<br><br>Tarkvaraaliidestuste stiil, kasutatakse päringute ja nende vastuste jaoks, suhtleb läbi Hüperteksti edastusprotokolli (*HTTP*). |
| **MQTT** (*Message Queuing Telemetry Transport*)<br><br>On sõnumite transpordiprotokoll, mis on kergesti implementeeritav ja töötab klient-maakler (ing. k. publisher-subscriber) printsiibil. |
| **JSON** (*JavaScript Object Notation*)<br><br>On lihtsustatud andmevahetustevorming, mis on loodud JavaScripti programmeerimiskeele süntaksi põhjal. |
| **Mock** (*Mock object*)<br><br>On simuleeritud objekt, mida kasutatakse vajalike andmete testeerimiseks. |
| **Tekstuur** (*Texture*)<br><br>On pilt, mis näeb välja nagu materjal või objekt. |
| **Blender**<br><br>3D objektide ning animatsioonide loomise tarkvara. |

## II. Kaasapandud failid

Visualisatsioon asub tööga kaasas olevas ZIP-failis „DeltaBuildingVisualization.zip”.

Visualisatsiooni lähtekood on kättesaadav GitLab repositooriumist, mis asub aadressil: https://gitlab.com/aleks96n/DeltaBuildingVisualization

# III. Python abiprogramm

Abiprogramm asub tööga kaasas olevas failis „TunniplaanAbiprogramm.py”.

# IV. Tunniplaan

Tunniplaan asub tööga kaasas olevas failis „Timetable.json”.

Käsitsi loodud tunniplaani fail asub tööga kaasas olevas failis „KasitsiTunniplaan.txt”.

## V. Ilmastiku andmed JSON formaadis


```
{
1.   "coord":{
2.     "lon":26.72,
3.     "lat":58.37
4.   },
5.   "weather":[
6.     {
7.       "id":300,
8.       "main":"Drizzle",
9.       "description":"light intensity drizzle",
10.      "icon":"09d"
11.
12.  ],
13.  "base":"stations",
14.  "main":{
15.    "temp":1,
16.    "pressure":986,
17.    "humidity":100,
18.    "temp_min":1,
19.    "temp_max":1
20.  },
21.  "visibility":2000,
22.  "wind":{
23.    "speed":2.6,
24.    "deg":60
25.  },
26.  "clouds":{
27.    "all":92
28.  },
29.  "dt":1522655400,
30.  "sys":{
31.    "type":1,
32.    "id":5015,
33.    "message":0.0022,
34.    "country":"EE",
35.    "sunrise":1522640243,
36.    "sunset":1522688233
37.  },
38.  "id":588335,
39.  "name":"Tartu",
40.  "cod":200
}
```


Selgitused:

Teenus tagastab linna koordinaate (read 1-4), ilma kirjeldust (read 5-12), sisemisi parameetreid (read 13, 31-33, 40), temperatuuri (read 15, 18-19), õhurõhku (rida 16), õhuniiskust (rida 17), nähtavust (rida 21), tuule informatsiooni (read 22-25), pilvesust (rida 26-28) ja linna parameetreid (read 34-36, 38-39)

## VI. Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina, **Andrei Voitenko**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
**Delta õppehoone keskkonna visualiseerimine**,
(*lõputöö pealkiri*)

mille juhendaja on Raimond-Hendrik Tunnel,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2018**