Tallinna Ülikool
Informaatika Instituut

# Google Earth API juhendmaterjali koostamine

Seminaritöö

Autor: Ronald Kaul
Juhendaja: Jaagup Kippar

Tallinn 2011

# Sisukord

## Sissejuhatus

Seminaritöö teemaks on Google Earth API ehk Google Earth´i programmiliidese tutvustamine läbi eesti keelse kasutusjuhendi. Google Earth API võimaldab kasutada Google Earth rakendust meile sobival veebilehel, ilma et oleks vaja midagi alla laadida, või kasutada Google Earth´i eraldiseisvat rakendust.

Google Earth on Google omanduses olev vabavaraline tarkvara, millega on võimalik vaadelda tervet maakera satelliit fotode abil, lisaks vaatlemisele on olemas ka geograafiline tugi, mis omakorda võimaldab väga täpselt määrata asukohti, teepikkuseid, kõrgust merepinnast ning palju muud. Olemas on aadresside süsteem ning ka vastav otsing, mis lubab meil mistahes maailmapunkti üles leida, kui teame selle koordinaate, või täpset aadressi. Mõne asukoha leidmiseks piisab ka seal asuva ehitise üldlevinud nimest, näiteks otsingusse „Tartu Ülikooli“ sisestades leiamegi ennast Tartu Ülikooli vaatlemas kaardil.

Seminaritöö peamiseks eesmärgiks on anda ülevaade kuidas Google Earth API töötab ning anda kasutajale juhised uue rakenduse loomiseks läbi autori poolt valitud näidete. Autor vaatleb lähemalt kaardi kuvamist ning seal ekskursioonide simuleerimist, koos näidetega kaardil kasutatavatest kohahoidjatest.

Eesmärgi saavutamiseks on kasutatud kasutusjuhendi loomist, koos selle testimisega potentsiaalsete kasutajate peal, ning seejärel tagasiside alusel juhendi parendamine ja vajadusel ka täiendamine vastavalt kasutajate soovitustele.

Juhendis vaatleme lähemalt ühe rakenduse näidet, ning lõpus saab juhendi kasutaja õpitu ka proovile panna tehes läbi 2 autori poolt koostatud ülesannet.

# 1 Juhend Google Earth API kasutamiseks

Käesolev juhend on mõeldud ülevaate saamiseks ja põhiteadmiste omandamiseks Google Earth API-st. Vaja läheb meil serveriruumi, kuhu saame faile üleslaadida, Google Earth rakendust ning soovitatavalt ka mingit tekstiredaktorit, kus oleks hea koodi kirjutada. Mina valisin tekstiredaktoriks Notepad++ nimelise programmi.

Google Earth API on mõeldud edasijõudnud kasutajatele, hõlmates samu võimalusi nagu Google Mapsis aga lisaks veel ka 3D tuge. Võimalik on 3D objektide kujutamine kaardil, teekonna simuleerimine, huvipakkuvate objektide märgistamine ja personaliseerimine, geograafiliste andmete kogumine ja palju muud, ning seda kõike brauseri aknas.

Google Earth API on toetatud järgnevatel platvormidel:

- Microsoft Windows (XP, and Vista)
    - Google Chrome 5.0+
    - Internet Explorer 7.0+
    - Firefox 3.0+
    - Flock 1.0+
- Apple Mac OS X 10.5 ja kõrgem (Intel)
    - Google Chrome 5.0+
    - Safari 3.1+
    - Firefox 3.0+

## 1.1 Unikaalse võtme genereerimine

Järgnevalt vaatame mida on vaja, et saada Google Earthi API tööle ka meid huvitaval lehel. Kõigepealt on meil vaja unikaalset võtit. Eelnevate versioonidega kaasnes sellega ka registreerimine, alates 3 versioonist pole see vajalik ja võtme saab genereerida Google Maps API family lehel:

http://code.google.com/apis/maps/signup.html

Kirja tuleb panna ka veebilehe nimi, kus rakendust hakatakse selle võtmega kasutama. Tähtis on see, et veebileht saaks kirja õigesti, kuna kui sisestada ainult näiteks veebi aadress:

http://www.mygooglemapssite.com

töötab see võti küll lehtedel:

http://www.mygooglemapssite.com

http://www.mygooglemapssite.com/maps

aga mitte veebilehtedel:

http://mygooglemapssite.com

http://host1.mygooglemapssite.com/

Seega soovitab Google kasutada veebilehe aadressi kuju: http://mygooglemapssite.com selleks, et tagada Google Earth API toimimine kõikidel antud domeeniga seotud veebilehekülgedel.

Kui oleme võtme ära genereerinud tuleks see kindlasti kusagile salvestada, kuna seda läheb hiljem kindlasti vaja.

## 1.2 HTML lehe loomine

Vaatame nüüd lähemalt, mis me koodi peame kirjutama. Kõigepealt teeme tühja html lehe kuhu hakkame lisama vajalikku koodi, et saada API meie lehel tööle. (Joonis 1.)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
      <head>
                  <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
            <title>Google Earth API proov</title>
      </head>
      <body>
                  <p></p>
        </body>
</html>
```

Joonis 1. Lihtne html

Nüüd lisame <*script*> tagid <*head*> tagide sisse, kus ABCDEF asemele tuleb panna võti mille enne genereerisime Google API lehel. (Joonis 2.)

```
<script type="text/javascript"
src="https://www.google.com/jsapi?key=ABCDEF"></script>
```

Joonis 2. Script tagid

Lisame veel ühed <*script*> tagid, kuhu alla hakkame me funktsioone lisama. (Joonis 3.)

```
<script type="text/javascript"></script>
```

Joonis 3. Script tagid 2

Järgmisena läheb viimasena loodud <*script*> tagide alla järgnev koodijupp (Joonis 4.), mis kutsub välja *„earth“* mooduli, mis võimaldab meil hiljem kasutada *google.earth* funktsiooni. Sulgudes olev 1 täpsustab API versiooni, seda saab ka veel täpsemalt kirja panna kasutades kuju „ 1.x“.

```
google.load("earth", "1");
```

Joonis 4. Google.load

Lisame <*body*> tagide sisse järgneva unikaalse id-ga DIV elemendi (Joonis 5.). Tähele tuleks panna, et *„map3d“* kasutame hiljem rakenduse väljakutsumiseks.

```
<div id="map3d" style="height: 400px; width:
600px;"></div>
```

Joonis 5. Uus div element

Nüüd hakkame funktsioone kirjutama. Funktsioonid lähevad kõik teisena loodud <*script*> tagi vahele. Meil läheb vaja 3 funktsiooni. Kõigepealt funktsioon millega loome uue eksemplari oma rakendusest (Joonis 6.). Sulgude sees on kõigepealt DIV element *„map3d“* mille sees edukal väljakutsumisel funktsiooni kuvatakse. Järgnevalt funktsioon „initCB“, mida rakendatakse siis kui uue eksemplari loomine läheb edukalt. Viimasena on sulgudes funktsioon failureCB, mida rakendatakse siis kui uue eksemplari loomine ei õnnestu.

```
function init() {
   google.earth.createInstance('map3d', initCB,
failureCB);
}
```

Joonis 6. Init funktsioon

Loome nüüd funktsiooni *„initCB“* (Joonis 7.), mis sisaldab endas kõike põhilist, et meie lehel kuvada Google Earth API-t. Funktsiooni sees luuakse uus muutuja *„ge“*, mis kutsub välja DIV tagi sees Google Earth´i akna, meetodiga *„setVisibility(true)“* tehakse see aken meile ka nähtavaks.

```
function initCB(instance) {
   ge = instance;
   ge.getWindow().setVisibility(true);
}
```

Joonis 7. Funktsioon initCB

Defineerime uue muutuja *„ge“* ka <*script*> tagide sees, kuhu lisame funktsioonid. (Joonis 8.)

```
var ge;
```

Joonis 8. Uus muutuja „ge“

Lisame ka funktsiooni „failureCB“ (Joonis 9.), mis kuvab meile vea korral veateate, mille järgi saab vajadusel viga otsima hakata.

```
function failureCB(errorCode) {
}
```

Joonis 9. Funktsioon failureCB

Viimasena läheb kirja Google'i enda funktsioon (Joonis 10.), mis tagab selle, et rakendust ei laadita enne, kui lehe enda DOM(*Document Object Model on platvormiülene ja keelest sõltumatu norm objektide esitamiseks ja nende omavaheliseks suhtlemiseks HTML, XHTML ja XML dokumentides. Osa DOM'i aspekte (näiteks tema elemente) võib adresseerida ja manipuleerida kasutatava programmeerimiskeele süntaksi sees. Dom'i avalik liides on täpsustatud tema Application Programming Interface (API) sees.l*) on ülesehitatud.

```
google.setOnLoadCallback(init);
```

Joonis 10. google.setOnLoadCallback

Nüüd peaks meil koos olema järgnev kood (Joonis 11.):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
      <head>
                  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">

   <title>Google Earth API proov</title>
   <script type="text/javascript"
src="https://www.google.com/jsapi?key=ABCDEFG">
   </script>
   <script type="text/javascript">
      var ge;
      google.load("earth", "1");

      function init() {
         google.earth.createInstance('map3d', initCB, failureCB);
      }

      function initCB(instance) {
         ge = instance;
         ge.getWindow().setVisibility(true);
      }

      function failureCB(errorCode) {
      }

      google.setOnLoadCallback(init);
   </script>

</head>
<body>
   <div id="map3d" style="height: 400px; width: 600px;"></div>
</body>
</html>
```

Joonis 11. Valmis kood

Hetkel see midagi muud meil ei tee kui näitab meie html aknas lihtsalt Google Earth`i akent.

## 1.3 KML faili loomine

Järgmisena vaatame midagi keerulisemat ja püüame lihtsa ekskursiooni teha. Selleks on meil vaja lisada koodi ja tekitada ka üks uus KML tüüpi fail, see hakkabki sisaldama meie ekskursiooni puudutavat infot.

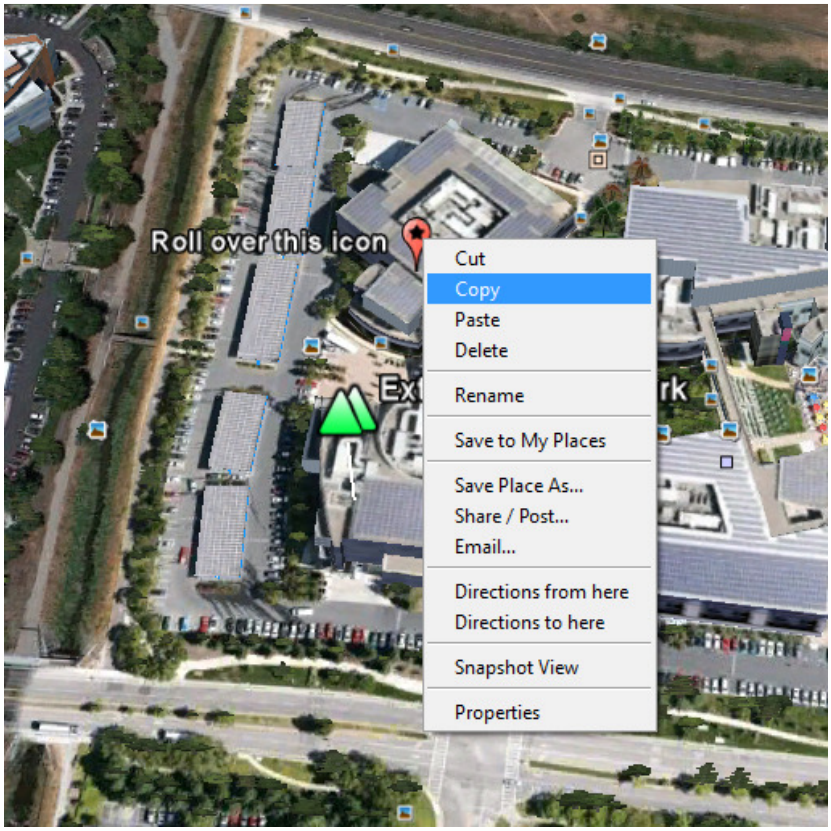
KML on XML standarditel baseeruv faili formaat, mida kasutatakse geograafiliste andmete kuvamiseks Earth brauserites: Google Earth, Google Maps ja Google Maps for Mobile.

Rohkem kml faili ja seal kasutatavate süntaksite kohta saab lugeda siit:

http://code.google.com/intl/et/apis/kml/documentation/kmlreference.html

KML faili tegemiseks on 2 moodust, kas kirjutada kõik käsitsi või ehitada antud näite puhul ekskursioon Google Earthis ja sealt kopeerida meid huvitavate kohahoidjate ja teekondade andmed ning moodustada nendest kml fail. Kasutame teist varianti, kuna see on oluliselt mugavam ja kiirem.

Andmete kopeerimine käib väga lihtsalt, tekitame Google Earth rakenduses näiteks *placemark'i* soovitud kohta, lisama sinna meid huvitava info, seejärel parem klikk *placemark'i* peal ning valime *copy*. Seejärel avame suvalise tekstiredaktori ja kopeerime kogu info sinna. (Joonis 12.)




Joonis 12. Google Earthi`st kopeerimine

Antud näite põhjal sain sellise koodi, nagu näha on kõik töö tegelikult minu eest juba ära tehtud ja kõik vajalikud tagid ja vormindus on juba olemas. (Joonis 13.)

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
        <name>KmlFile</name>
        <Style id="globeIcon">
                <IconStyle>
                        <Icon>

        <href>http://maps.google.com/mapfiles/kml/pal3/icon19.png</href>
                        </Icon>
                </IconStyle>
                <LineStyle>
                        <width>2</width>
                </LineStyle>
        </Style>
        <Placemark id="Estoniateater">
                <name>Estonia teater</name>
                <description>Rahvusooper Estonia</description>
                <LookAt>
                        <longitude>24.75104042359311</longitude>
                        <latitude>59.43431762708924</latitude>
                        <altitude>0</altitude>
                        <heading>-0.4047094173224636</heading>
                        <tilt>38.2212487140112</tilt>
                        <range>496.4783392307506</range>
                        <altitudeMode>relativeToGround</altitudeMode>

        <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
                </LookAt>
                <styleUrl>#msn_ylw-pushpin</styleUrl>
                <Point>
                        <altitudeMode>clampToGround</altitudeMode>

        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>

        <coordinates>24.75111555506787,59.43457909979984,0</coordinates>
                </Point>
        </Placemark>
</Document>
</kml>
```

Joonis 13. Kml faili sisu

Vaatame nüüd natuke lähemalt saadud koodi.

- Esimesel real on meil XML päis, see peab alati olema esimene rida
- Järgmisel real on nimeruumi deklaratsioon(namespace declaration ing. keeles), alati 2 rida.
- Algavad *Document* tagid kus sees on kogu info punkti kujunduse ja asukoha kohta

- *Style* tagide sees on punkti kujundust kirjeldavad parameetrid, ikoon(*IconStyle*), joone paksus ikoonist maapinnani(*LineStyle*)
- *Placemark* tagide sees on info antud punkti kohta, punkti nimi(*name*), kirjeldus(*description*) ning *LookAt* tagid määravad ära kaamera asetuse punkti kohal(*longitude* ja *latitude*), kõrguse(*altitude*), suuna(*heading*), kalde(*tilt*) ja kauguse(*range*).
- *StyleUrl* määrab ära placemark'i ikooni.
- Järgmisena näeme *Point* tage, seal sees on siis tegelikult kõige tähtsam info, ehk meid huvitava punkti geograafiline asukoht(*coordinates*) ja ka näiteks see kuidas punkti kõrgust peaks arvutama(*altitudeMode*), mis näites on seatud suhteliseks maapinna suhtes(*relativeToGround*).

Järgnevalt vaatame kuidas võiks teha virtuaalset ekskursiooni edasi. Mina talitasin nii, et lisasin veel huvipakkuvaid punkte ning seejärel kasutasin Google Earthi enda võimalusi ja salvestasin ekskursiooni kasutades „*Record a Tour*" funktsiooni. (joonis 14.)


Joonis 14. Record a Tour

Tegemist on põhimõtteliselt *screen capture* tüüpi lisaga, mis lihtsalt salvestab kõik tegemised mis ekraanil toimuvad ja hiljem on võimalik seda tagasi mängida. Seejärel salvestasin ekskursiooni ja jällegi kopeerisin kogu info vast loodud ekskursiooni peale klikkides parema hiirenupuga ja lisasin gx:tour tagide sees oleva koodi osa <*document*> tagide sisse olemasolevas kml failis.

Järgnevalt vaatame ühte juppi saadud ekskursiooni koodist. (Joonis 15.)

```
<gx:Tour>
      <name>Untitled Tour</name>
      <gx:Playlist>
            <gx:FlyTo>
                  <LookAt>
                        <longitude>24.75679440074899</longitude>
                        <latitude>59.4375986053527</latitude>
                        <altitude>0</altitude>
                        <heading>6.973705273199008</heading>
                        <tilt>0</tilt>
                        <range>1224.128098905278</range>
                        <altitudeMode>relativeToGround</altitudeMode>

      <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
                  </LookAt>
            </gx:FlyTo>
            <gx:Wait><gx:duration>7.679760116729995e-007</gx:duration>
</gx:Wait>
```

Joonis 15. gx:Tour sisu

- Kõigepealt näeme <gx:tour> tagi, nende tagide sees on kogu ekskursiooni puudutav info.
- <*gx:Playlist*> sisaldab endas punkte mida ekskursiooni ajal külastatakse.
- <*gx:FlyTo*> kasutatakse ühest punktist teise liikumiseks, selle all on juba tuttavad tagid, mis määravad ära kaamera asukoha
- <*gx:Wait*> ja selle sees asuv <*gx:turation*> määravad ära pausi pikkuse enne kui uue punkti juurde minnakse

## 1.4 KML faili väljakutsumine HTML-is

Nüüd on meil olemas küll kogu info selle kohta. kus punktid on ja kuidas kaamera liikuma peaks aga vaja oleks ka veel see kõik kenasti välja kutsuda, ning ka kuidagi kontrollida.

Lisame uued <*script*> tagid. Nende sees sisaldub viide *javascript* failile (Joonis 16.), millega kammitakse kml fail läbi ja otsitakse sealt meie ekskursiooni jaoks vajalik info, seda on kindlasti siis vaja kui kml fail sisaldab lisaks ekskursioonile ka muid objekte näiteks Placemarke või kasutatakse <*document*> või <*folder*> tage.

```
<script src="http://earth-api-
samples.googlecode.com/svn/trunk/lib/kmldomwalk.js"
type="text/javascript"> </script>
```

Joonis 16. Kmldomwalk.js

Järgmisena täiustame funktsiooni *initCB*, funktsioon *getNavigationControl()* (Joonis 17.) võimaldab meil näha kaamera asukoha kontrollimiseks vajalikku kasutajaliidest. Ning „var href“ (Joonis 17.) näitab ära meie kml faili asukoha, see peab kindlasti olema *url´i* ehk veebiaadressi kujul..

```
   function initCB(instance) {
      ge = instance;
      ge.getWindow().setVisibility(true);
      ge.getNavigationControl().setVisibility(ge.VISIBILITY_SHOW);

      var href = var href = 'http://www.tlu.ee/~rk/earth/newtour.kml';
      google.earth.fetchKml(ge, href, fetchCallback);
   }
```

Joonis 17. Uus funktsioon getNavigationControl()

On olemas kolm meetodit kml faili laadimiseks meie rakendusse.

- *KmlNetworkLink* laeb KML või KMZ faili spetsiifilise *URL* aadressi kaudu. Saadud kml fail on võimalik seejärel lisada rakendusse nagu tavaline objekt, kasutades käsklusi *ge.getFeatures().appendChild().*
- *fetchKml* lisab samuti KML faili kasutades URL aadressi aga tagastab *kmlfeature* objekti, objekti *DOM*'i on võimaldatud juurdepääs ja vajadusel ka uuendamine enne kui see lisatakse rakendusele külge.
- *parseKml* võtab KMl stringi ja tagastab samuti *kmlfeature* objekti, nagu ka *fetchkml*'iga on ka siin tagatud juurdepääs *DOM*'ile.

Meie kasutame oma näites *fetchKml* meetodit (Joonis 18.), kuna see võimaldab meil kasutada eraldiseisvat faili ja samuti on tagatud juurdepääs veebilehe DOM'ile. Lisame uue funktsiooni kml faili kasutamiseks.

```
        function fetchCallback(fetchedKml) {
        //kui faili ei leita teavitatakse sellest
      if (!fetchedKml) {
        setTimeout(function() {
          alert('Bad or null KML');
        }, 0);
        return;
      }
        //lisame kml faili earth muutujasse
        ge.getFeatures().appendChild(fetchedKml);
        }
//käime läbi kml faili, et leida tour tagid, lisame need muutujasse
„tour“
          walkKmlDom(fetchedKml, function() {
             if (this.getType() == 'KmlTour') {
                tour = this;
                return false;
             }
          });
         }
```

Joonis 18. funktsioon fetchCallback

Järgmisena lisame funktsioonid ekskursiooni animatsiooni kontrollimiseks. (Joonis 19.)

```
function enterTour() {
  if (!tour) {
    alert('No tour found!');
    return;
  }
  ge.getTourPlayer().setTour(tour);
}
function playTour() {
  ge.getTourPlayer().play();
}
function pauseTour() {
  ge.getTourPlayer().pause();
}
function resetTour() {
  ge.getTourPlayer().reset();
}
function exitTour() {
  ge.getTourPlayer().setTour(null);
}
```

Joonis 19. Ekskursiooni animatsiooni funktsioonid

- Funktsioon enterTour viib meid ekskursiooni sisse, alguspunkti. Selleks kasutatakse muutujat „tour“, mis omakorda tuleb kml failist.
- Funktsioonid playTour mängib ekskursiooni.
- Funktsioon pauseTour paneb pausi peale ekskursiooni.
- Funktsioon resetTour viib meid ekskursiooni algusesse tagasi.
- Funktsioon exitTour väljub ekskursioonist.

Ning viimasena lisame nupud <*body*> tagide vahele, millega me hakkame oma loodud ekskursiooni kontrollima. (Joonis 20.)

```
<div id ="controls">
  <input type="button" onclick="enterTour()" value="Sisene ekskursioonile"/>
  <input type="button" onclick="playTour()" value="Mängi"/>
  <input type="button" onclick="pauseTour()" value="Paus"/>
  <input type="button" onclick="resetTour()" value="Stop"/>
  <input type="button" onclick="exitTour()" value="Välju"/>
</div>
```

Joonis 20. Ekskursiooni kontrollimise nupud

Nüüd peakski meil valmis olema ekskursioon, kus saame tutvustada näiteks oma kodulinna, vaatamisväärsuste asukohti või niisama huvitavaid kohti külastada.

Minu valminud näidet on võimalik uurida veebiaadressil:

http://www.tlu.ee/~rk/earth/

Tekstiversioonid failidest on olemas veebiaadressil:

http://www.tlu.ee/~rk/earth/koodid/

## 1.5 Inglisekeelsed kasutusjuhendid

Olemas on terve hulk erinevaid inglise keelseid juhendeid, asuvad nad Google Earth API *Developers guide'is*: http://code.google.com/apis/earth/documentation/index.html
Tegemist on väga korralike juhenditega , kus kood on põhjalikult lahti seletatud ja lisatud on nii koodinäidised kui ka valmis näidete veebiaadressid. Olemas on näiteid järgnevatest valdkondadest:

1. **Placemarks** - seletab lähemalt *placemark´ide* kohta, kuidas kasutada, mis lisad on olemas ning mis võimalused on nende kujundamiseks.
2. **Balloons** - eesti keeles teatud kui tekstimullid, kuidas neid kasutada, mis võimalused on neid kujundada ning kuidas kasutada html-i nende sees.
3. **Geometries & Overlays** - ruumilised kujundid Google Earth`is, nende loomine lahtiseletatult ning Google Earth´i maapinnale projitseeritavad pildid ning nende kasutamine.
4. **Camera Control** - kaamera kontrollimine Google Earth`is
5. **Layers & Controls** - tutvustab erinevaid tasandeid mis on olemas Google Earthis`is, näiteks Atmosfäär ja Päike, 3D majad, Puud, 3D maastik jne.
6. **Time** - kuidas reaalajas vaadata päikese asukohta, ning varjude teket kaardil, samuti kuidas pääseda ligi kaardi arhiividele ja vaadata vanemaid satelliitpilte mingist piirkonnast.
7. **Ocean** - vaadeldakse lähemalt veekogusid ning seda kuidas lisada kujundeid veekogude peale, põhja ja sisse.
8. **Sky, Mars, & Moon** – kuidas vaadata taevast, Marsi ja Kuud Google Earth`is.
9. **Touring** - seletab lähemalt ekskursioonide tegemisest, importimisest ning nende mängimisest.
10. **Events** - kuidas kasutada erinevaid sündmusi Google Earth`is, kuidas neid välja kutsuda näiteks hiirega, ning kuidas neid hiljem eemaldada.
11. **Accessors** - lisad, erinevad lisakäsud *url´i* ja kml failiga töötamiseks.
12. **Object Containers** – erinevad konteinerid *array* elementide hoidmiseks
13. **KML** - geograafilise info hoidmiseks kasutatav failitüüp, milleks kasutatakse, kuidas importida oma rakendusse ning mis võimalused selle kasutamiseks.
14. **Options** – tutvustab erinevaid sätteid Google Earth`is, näiteks mõõtühikud, kasutatav keel ja 3D maastiku võimendamine.
15. **Debugging** – tutvustab erinevaid brauserite *debugger`eid* ning kuidas neid kasutada.

## 1.6 Kokkuvõte

Google Earth ja Google Earth API võimalused on väga suured, võimalik on joonistada kaardile 3D objekte ja neid ka liikuma panna, simuleerida erinevaid teekondi, koguda ja esitada erinevat infot maastiku kohta ning isegi teha interaktiivseid geograafilisi rakendusi, mida saab mugandada ka näiteks mängudeks.
Üheks selliseks on Geo Whiz, mängu eesmärgiks on otsida üles küsitavad punktid, liikudes kaardil nende peale. Geograafia tunnid on kindlasti tänu sellele palju huvitavamad.
Seda saab vaadata allolevalt lingilt:
http://earth-api-samples.googlecode.com/svn/trunk/demos/geowhiz/index.html

Kasutades praegu õpitud teadmisi saame juba edasi luua igasugu rakendusi Google Earth`i baasil, kõigi võimaluste väljatoomiseks peaks koostama kõvasti mahukama materjali aga nendest baasteadmistest peaks piisama, et iseseisvalt edasi minna.

# 2 Kasutajate tagasiside

Kasutajate tagasiside põhjal, mis toimus peamiselt meili teel, tuli välja ka paar suurt puudust, mis said lõplikus juhendis ära parandatud. Juhendit sai lihtsamaks ja arusaadavamaks tehtud, spetsiifilised sõnad teksti sees lahti seletatud ning vajadusel ka lause struktuure muudetud, et olla kindel et tekst on kõigile üheselt mõistetav. Samuti ei sobinud kõikidele esialgne viitamise süsteem, mis tegelikult oli puudulik, kuna see kujutas endast seda, et koodijupi näidis oli pandud vastava lõigu lõppu aga see tekitas segadust. Kõik pildid ja koodinäited said ilusti endale nimed külge ja teksti sees ka nendele vastavad viidad õigesse kohta.

Tulid välja ka mõned tehnilised puudused. Esiteks see, et kõikidel ei olnud Google Earth plugin´i brauseri jaoks arvutis ja nende jaoks sisaldas juhendi katsetamine ka seda, et pidid selle alla tõmbama oma arvutisse. Kuigi juhendi kohaselt ei oleks nad midagi pidanud installeerima ega alla tõmbama.

Samuti esines kohati probleeme Firefoxi veebibrauseriga, mis ei tahtnud alati koostööd teha. Võis juhtuda, et pannes rakenduse Firefoxi brauseris tööle, jooksis mingi aja pärast kogu rakendus kokku. Põhjuseks oli see, et *script* jäi lihtsalt tsüklisse kinni ja seda ei olnud võimalik peatada. Internet Explorer samas töötas eeskujulikult ja sellega probleeme ei esinenud.

Omaette teema on ka interneti kiirusega, kuna kaardid võetakse Google Earthi serverist ja nende allalaadimine rakendusse võtab aega ei ole see just kõige meeldivam kõikidele kasutajatele. Muidugi on interneti kiirused pidevalt kasvanud ja laialdaselt on levinud juba väga kiire internet on siiski olemas ka kasutajaid, kes olude sunnil peavad kasutama võrdlemisi aeglast internet ja nende jaoks on kaardi laadimise ootamine suhteliselt tüütu.

# 3 Kokkuvõte

Antud seminaritöö eesmärgiks oli tutvustada Google Earth API-t läbi eesti keelse kasutusjuhendi, milles tehti läbi põhilised sammud rakenduse esialgseks edukaks käivitamiseks ning tutvustati ka mõningaid funktsioone läbi näidete.

Kasutusjuhendi loomisel kasutati inglise keelset materjali, mis leidub Google Earth API Developers Guide veebilehel, ning seejärel testiti saadud eesti keelset juhendit kasutajate peal. Kasutajate tagasisidet arvesse võttes pidevalt täiustati ja edendati juhendit selliseks, et oleks tagatud:

- Üheselt arusaadav tekst
- Lihtne loetavus
- Süstematiseeritud viitamine
- Pidev koodi kontrollimise võimalus

Testimise käigus tulid välja ka mõned kitsaskohad, näiteks võib Firefoxis Google Earthi rakendus kokku joosta, ning sõltuvalt interneti kiirusest võib olla Google Earthi kasutamine brauseris tülikas. Seega päris igal pool ei ole otstarbekas nii interaktiivset ja detailset kaarti kasutada, piisab ka palju lihtsamast kaardist kasvõi pildikujul.

Autor leiab, et selle juhendiga sai kasutajatele selgeks tehtud Google Earth API põhi omadused ja tõed ning selle abil saab juba edaspidi iseseisvalt katsetada rakenduse kõiki teisi funktsioone.

## Kasutatud allikad

Google Earth API Developer's Guide. Viimati vaadatud 28.10.2011
http://code.google.com/intl/et/apis/earth/documentation/

KML Tutorial. Viimati vaadatud 28.10.2011
http://code.google.com/intl/et/apis/kml/documentation/kml_tut.html

Google Earth API inglisekeelsed juhendid. Viimati vaadatud 28.10.2011
http://code.google.com/apis/earth/documentation/index.html

HTML kontrollimine. Viimati vaadatud 14.11.2011
http://validator.w3.org/

HTML standard. Viimati vaadatud 14.11.2011
http://www.w3schools.com/html/html_doctype.asp

# LISAD

## Ülesanded

Järgnevalt paar ülesannet iseseisvaks lahendamiseks. Näidete valmisversioonid on olemas veebilehel: http://www.tlu.ee/~rk/earth/ylesanded/

### Ülesanne 1

Vaatame rakendust „Kohaleidja“, milles kasutame *ClientGeocoder* funktsiooni ning millega on võimalik tutuvuda veebiaadressil:

http://www.tlu.ee/~rk/earth/ylesanded/kohaleidja_alg.html

Täiendame andtud rakendust nii, et saaksime valida 5 sihtkohta *drop down* menüüst, ning seejärel nupu vajutusega valikut kinnitades nende peale liikuda kaardil.

Sihtkohtadeks võtame:

- Stanfordi Ülikool
- Tallinna Tehnika Ülikool
- Tartu Ülikool
- Harvard
- Tallinna Ülikool

### Ülesanne 2

Täiendame esimeses ülesandes saadud rakendust nii, et iga koha juures, mida saame *drop-down* menüüst valida oleks olemas ka *placemark*, mis lühidalt kirjeldab seda kohta.

### Lahendused:

Esimese ülesande lahenduse kood on olemas veebiaadressil:

http://www.tlu.ee/~rk/earth/ylesanded/kohaleidja_dropd.html