Tallinna Ülikool
Digitehnoloogiate Instituut

PlayCanvas keskkonnas 3D mängu loomine

Seminaritöö

Autor: Heikki Laidinen
Juhendaja: Martin Sillaots

Autor: ..... ,, ... ,, 2016
Juhendaja: ..... ,, ... ,, 2016

Tallinn 2016

Sisukord

Mõisted

Angular Damping - objekti keerlemise aeglustumine

Angular Factor - lineaarse keerlemise sujuvus

Collision - kokkupõrge

Cubemap - pildiseerija, mida kasutatakse objekti kujutamisel

Iframe - HTML dokument, kus kuvatakse mingit teist veebilehte

Linear damping - objekti liikumise aeglustumine

Linear factor - lineaarse liikumise sujuvus

MAMP - jooksutab veebilehti lokaalses masinas

Mängumootor - mängumootor on süsteem, mis on kavandatud 2D või 3D videomängude loomiseks ja arendamiseks.

Particle system - kogum objektidest

Plane - tasapind

Rigid body - füüsiline(tahke) keha objektil

Skybox- mängumaailma ümbritsev taevas

UI - kasutaja liides

WebGL - WebGL(*Web Graphics Library*) on Javascripti API, mis võimaldab Javascripti renderdada(visualiseerida) 3D ja 2D graafikat.

Sissejuhatus

Peamiselt on mänge loodud C#, C++ ja Java programmeerimis keeles, kuid WebGL väljatulekuga lisandus võimalus luua mänge puhtalt javascripti abiga ning 3D-d kasutada vabalt teiste HTML elementidega. WebGL on Javascripti API*(Application Programming Interface),* mis võimaldab Javascripti renderdada(visualiseerida) 3D ja 2D graafikat. Esimene WebGL versioon tuli välja 3. märtsil 2011, mis lubas 3D graafika kasutamist HTML5 lehtedel ilma lisamooduli kasutamiseta. (Hirshon)

PlayCanvas on üks neist 3D mängumootoritest, mis kasutab WebGL'i ning on maailma esimene pilvepõhiline mänguarenduse platvorm. (Evans, 2014) Lisaks PlayCanvasele on veel Three.js, Phaser, Thurbulenz ja veel mõningad. PlayCanvas erineb nendest oma pilveteenuse ja veebiredaktoriga, mis muudab mänguarendamise veelgi lihtsamaks.

Teema valis autor isiklikust huvist PlayCanvas keskkonna ja mängude loomise vastu. Lisaks soov tutvustada seminaritöö lugejatele ja mängutegijatele uuenduslikuma lähenemisega mängumootorit, mis võib lihtsustada mängude tegemist. Autori arvates eelnevad mängude tegemise programmid näiteks Unity on algajate jaoks üsnagi keerukad. PlayCanvas tõmbas autori tähelepanu just selle lihtsuse pärast, et sellega on võimalik teha mäng võimalikult väheste vahenditega. Töö eesmärk on anda ülevaade PlayCanvase keskkonnast ning tutvustada selle võimalusi näite mängu abil.

Töö esimeses osas antakse ülevaade PlayCanvasest, selle õppematerjalidest, keskkonnast, uue projekti tegemisest, olemasoleva projekti avalikustamisest. Töö teises osas tehakse PlayCanvase labürindi näitemäng esimese mängija vaates, mille eesmärk on tutvustada PlayCanvase 3D mängu tegemise objekte, objektide omadusi ning nende lisasid.

1 Ülevaade PlayCanvasest

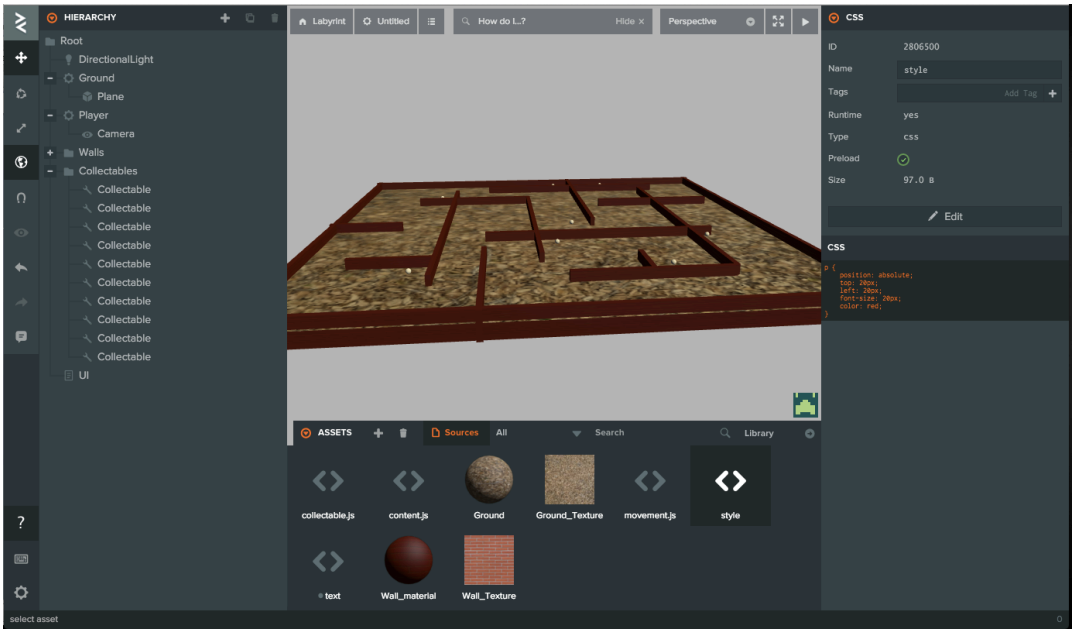
PlayCanvast kasutades saab teha 3D HTML ja WebGL mänge, reklaame ning demosid toodetele. (PlayCanvas) PlayCanvase mängumootor muutus kõigile vabalt kättesaadavaks 4. juulil 2014 MIT litsentsi alusel. (Evans, 2014) MIT litsentsi puhul võib tarkvara vabavaraliselt kasutada, muuta, kasutada äri eesmärgiliselt, ainuke kohustus on lisada tarkvara autori *copyright*. See andis võimaluse kõigil luua WebGL põhjal mänge, mida saab mängida veebibrauseris või mobiilses brauseris. Võimalik on samuti mobiili versioonina mänge eksportida äpi poodidesse. Hetkel on võimalik mobiiliversiooni mängust üles laadida *iOS App Store'i* või *Google Play'sse*. (Nyman & Eastcott, 2014)

PlayCanvas mängumootor lubab kasutada põhilisi mängu komponente, mis on vajalik kõrge kvaliteediga mängu tegemiseks nagu näiteks:

- Graafilised elemendid - vajalikult mängu visuaalsuse jaoks.
- Füüsikalised elemendid - võimaldavad 3D maailmas liikumise ning mängumaailmas füüsikalisi omadusi kasutada. Näiteks objekt ei saa läbi seina liikuda.
- Animatsioonid - kasutada animatsioone käte ja jalgade liigutamiseks tehes liikumise realistlikumaks.
- Audio – võimaldab kasutada 2D ja 3D audio efekte.
- Sisendseadmed – hiire, klaviatuuri, puutetundlikkuse ja mängupuldi tugi.

(Evans, 2014)

PlayCanvasel on oma veebiredaktor(vt Joonis 1), mis töötab veebilehitsejas. Veebiredaktoris saab lihtsalt nupu vajutamiste ning tirimisega tekitada 3D maailma objekte ning objektidele füüsikalisi omadusi anda. Lisaks objektile lisada animatsioone, heli ning funktsioone. Objektidele ümbrust (valgustus, *skybox*, udu) lisada. Samuti tekitada erinevaid stsenaariume. PlayCanvasel on tasuta versioon ning erineva astmega tasulised versioonid.



Joonis 1. Veebiredaktori vaade

Tasuta versioon

Tasuta versiooni saavad kõik kasutada. Limiteeritud on projektida arv (maksimaalselt 2 projekti), andmemaht (maksimaalselt 200MB kokku). Tasuta versiooni puhul kõik tehtavad projektid on avalikud. (PlayCanvas)

Pro versioon

Pro versioon on mõeldud rohkem üksikisikutele või siis väiksematele firmadele, mis teenivad aastas vähem kui 100 000 dollarit (~91 400 eurot). Pro versiooniga saab teha privaatseid projekte, mida tasuta versiooniga teha ei saanud. Privaatseid projekte võib olla 5-20, olenevalt, mis astme Pro versioon on. Samuti on suurem andmemaht saadaval (500MB - 10 000MB), lisaks saab olla rohkem tiimiliikmeid (5-20 liiget). Pro versiooniga lisandub ka iOS eksport võimalus. Kuutasu on 15 - 60 dollarit kuus, olenevalt, mis astme Pro versiooniga tegu. (PlayCanvas)

Org versioon

Org versioon on mõeldud organisatsioonidele. Privaatsete projektide kogus, andmemahu suurus, tiimiliikmete arv ning tingimused on samad, mis Pro versioonil. Org versiooniga lisandub e-maili tugi, laadimislehe muutmis võimalus, projektide importimise ja eksportimise võimalus. Kuutasu on 100 - 400 dollarit kuus, vastavalt, mis astme Org versiooniga tegu. (PlayCanvas)

1.1 PlayCanvase õppematerjale

Veebis leiduvad PlayCanvase õppematerjale on väga vähe ja need olemasolevad on PlayCanvase enda tehtud. Autor on näite mängu puhul samuti PlayCanvase õppematerjali õpitust lähtunud.

http://developer.playcanvas.com/en/tutorials/ - PlayCanvas enda tekstipõhine õppematerjal, hõlmates kõike põhilist, mis on alustuseks vajalik teada. Õppematerjal on jaotatud kolmeks erinevaks tasemeks, arvestades õppija enda hinnangulist taset (algaja, kesktase, edasijõudnud). Iga õppematerjali teema juures on *iframe* abil töötav visuaalne näide. Autor soovitab seda eelkõige õppimisel kasutada.

https://www.youtube.com/user/playcanvas/videos - PlayCanvase *youtube* leht, kust leiab mitmeid mõne minutilist õpetlike videoõpetusi. Videoõpetustes näidatakse visuaalselt ekraanivideo abil, kuidas veebiredaktorit kasutada soovitud tulemuse saamiseks.

1.2 PlayCanvases mängu loomise etapid

Mängu loomise PlayCanvases võib jaotada mitmeks etapiks. Esimeseks etapiks on kasutaja tegemine ning projekti loomine. Seejärel saab sisuliste tegevuste tegemisega nagu objektide loomisega ning objektidele *skriptide* ja materjalide lisamisega, millele keskendutakse töö teises teemas läbi näite mängu tegemise. Viimaseks etapiks on loodud projekti avalikustamine.

Projekti alustamine

Esiteks tuleb end registreerida PlayCanvas keskkonnas. Sisselogimisel tehaksegi kohe esimene projekt ning suunatakse *editor'i*. Sinna on kõige esimene projekt tehtud, millele on lisatud mõned objektid koos tekstuuride ja skriptidega. Käesoleva töö jaoks kasutatakse uut eelseadistusteta projekt.

Projekti avalikustamine

Projekti avalikustamiseks peab olema sisselogitud ning minema *PROJECTS* lehele, kus tulevad ette olemasolevad projektid. Olemasolevate projektide seast valida avalikustamiseks mõeldud projekt ja minna selle projekti lehele(vajutades projekti nime peale). Seejärel avaneb lehe ülemises osas vaade, kus tuleb teha hiireklikk *PUBLISH* peale. Seejärel antakse kaks valikut publish ning download.

Publish valmisel tuleb ette väike aken, kuhu peab sisestama sobiliku äpi nime, seejärel valima stseeni ning klikivajutuse tegema *publish now* nupule. Peale nupu vajutus avalistatakse see PlayCanvas lehel, sellest antakse teada ka peale nupu vajutust ning teates sisaldub äpi aadress, kus see asetseb. *Iframe* abil võimalik seda ka enda veebilehel näidata.

Download valimisel tuleb ette väike aken, kuhu peab sisestama sobiliku äpi nime, seejärel valima stseeni ning klikivajutuse tegema *Web* nupule. Valikus oli *Web* ja *iOS*, kuid *iOS* on võimalik alates PlayCanvas kasutaja Pro litsentsist. Alla laetakse .zip kaust, kus on äpifailid. Lahtipakkimisel saab kausta panna enda veebilehele või jooksutada MAMP (sobib ka muu alternatiivne veebiserver) veebiserveris.

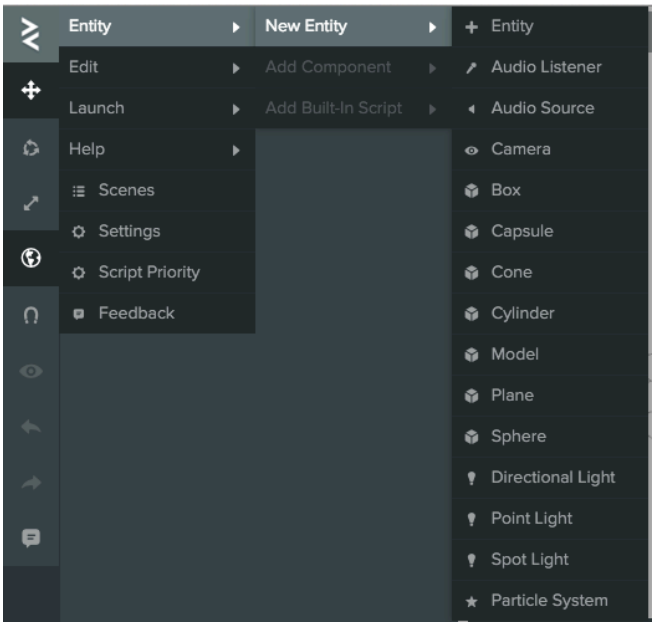
1.3 Veebiredaktori keskkond

Veebiredaktor on PlayCanvase üks olulisemaid kohti, kus tegeletakse mänguloomise kõige olulisema tegevusega ehk 3D maailma ja sealsete objektide loomisega. Veebiredaktorisse saab minna, kui liikuda veebilehitsejas kindla projekti alla ning klikivajutus teha *EDITOR* nupule. Peale klikivajutust tuleb veebiredaktori vaade(vt Joonis 1).

Menüü ja tööriistariba

Menüü avaneb, kui teha klikivajutus üleval vasakul olevale nupule, kus kujutatud kahte noolt(vt Joonis 2). Menüü sisaldab kõiki käsklusi, mida saab stseenil kasutada (PlayCanvas).

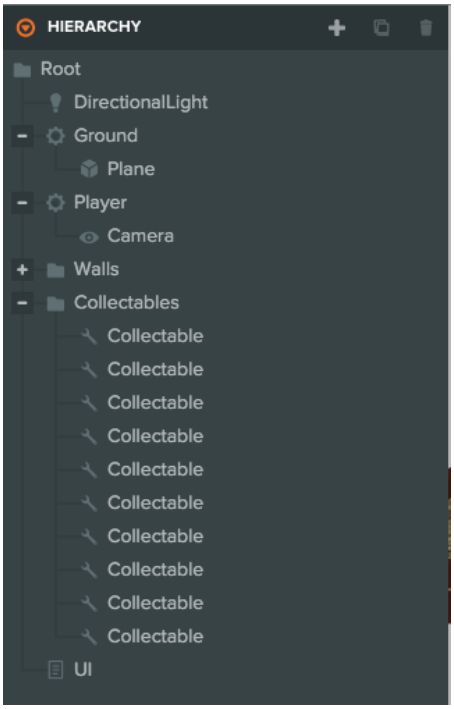
Tööriistaribal on mõned põhilised käsklused(vt Joonis 2). Samuti on abi nupp ja kiirklahvide nimistu nupp.



Joonis 2. Veebiredaktori menüü ja tööriistariba

Olemi hierarhia

Olemi hierarhia on kujutatud puukujulisena ja see koosneb stseeni olemitest, milleks võivad olla kaamera, valgutus, mudelid ja heli(vt Joonis 3). Hierarahia alamobjektid pärivad enda ülemobjektilt omadusi. Näiteks mitmele sarnasele objektile on niiviisi lihtsam lisada ühiseid omadusi.



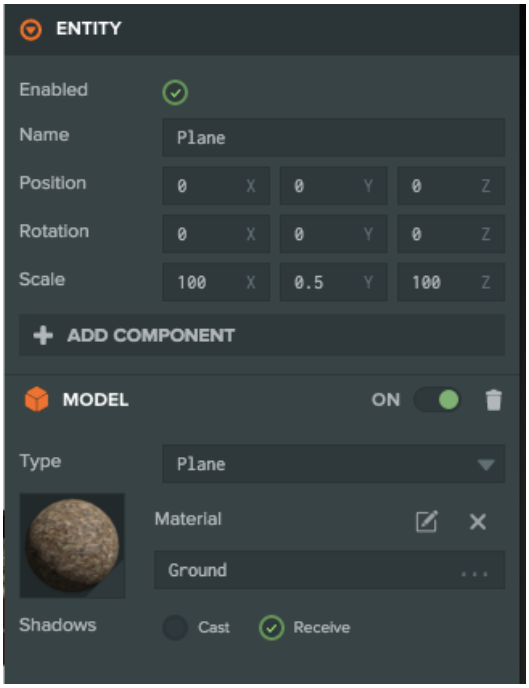
Joonis 3 Veebiredaktoris olemi hierarhia

Inspektor

Inspektoris saab vaadata olemi täpsemaid parameetreid(vt Joonis 4). Olemi parameetriteks võivad olla füüsikalised omadused(positsioon, rotatsioon, skaala). Olemile saab lisada komponente, komponentideks võivad olla:

- animatsioonid
- heli kuularid
- väljundheli
- kaamera
- valgus
- kokkupõrge
- mudel
- *Rigid Body* - keha
- osakeste süsteem(*Particle System)*
- skript

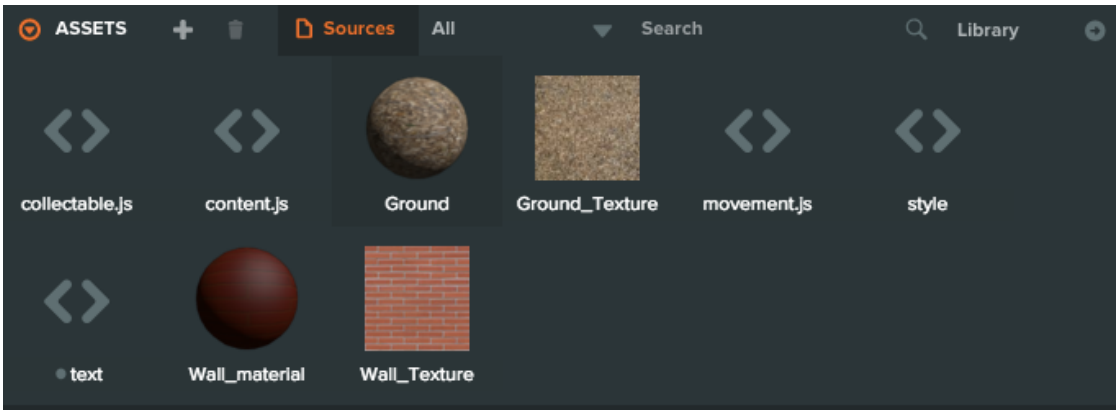
Minnes hiiregCa omaduse peale avaneb vasakule väike kast, kasti ilmub seletav osa omaduse kohta link PlayCanvase API viitele.



Joonis 4 Veebiredaktori inspektor

Ressursid(*Assets)*

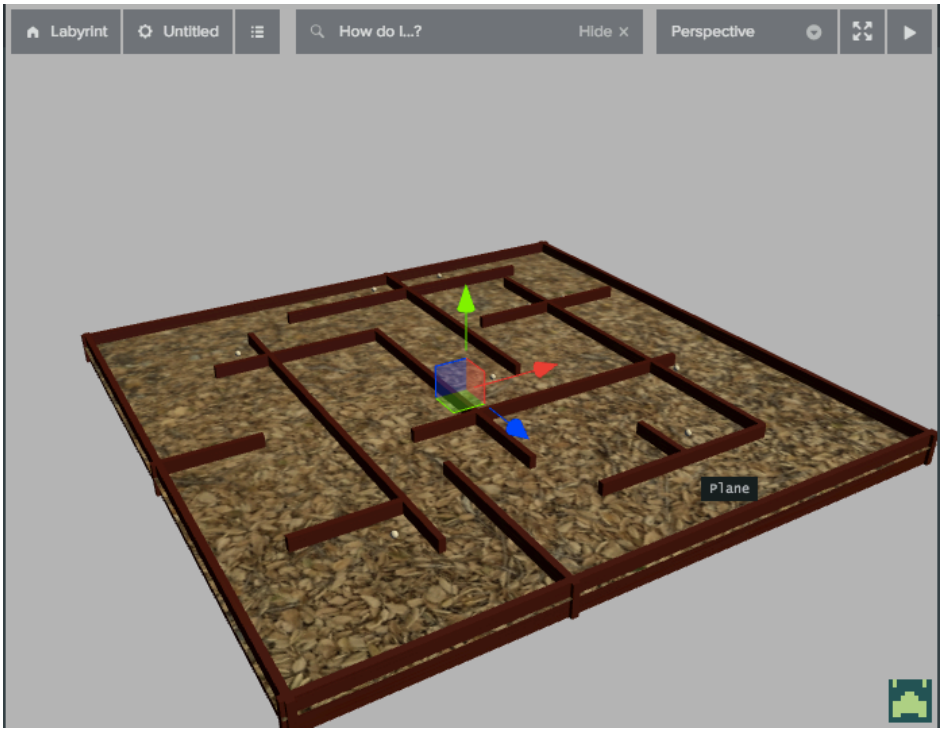
Ressurside all kuvatakse erinevaid tekstuure, materjale, objekte, skripte, heli, animatsioone, *cubemap*'e, html ja css faile, mida saab hiljem objektide külge panna. Kogumiku(*Library)* nupule vajutades suunatakse ressursi kogumiku lehele, kust saab olemasolevaid ja tasuta saadavaid ressursse allalaadida projekt. Arvutist failide lisamiseks tuleb need kogumikesse ülesse laadida. Ressursside kasutamisel tuleb need tirida sobivasse kohta. Objektid puhul tuleb tirida vaateaknasse.



Joonis 5 Veebiredaktori ressurssid

Vaateaken

Vaateaknas kuvatakse praegune olukord stseenist (vt Joonis 6). Vaateaknas saab objekte lähemalt vaadata erineva nurga alt ning nende suurusi, positsiooni ja pöördenurka muuta. Muuta saab ka stseeni, kui neid on rohkem kui üks. Kaamera vaatenurka on algselt määratud *perspective*, seda saab enda soovil muuta. Valikus on ka nurga alt objekti näitamine pealt, alt, ja kõrvalt.



Joonis 6 Veebiredaktori vaateaken

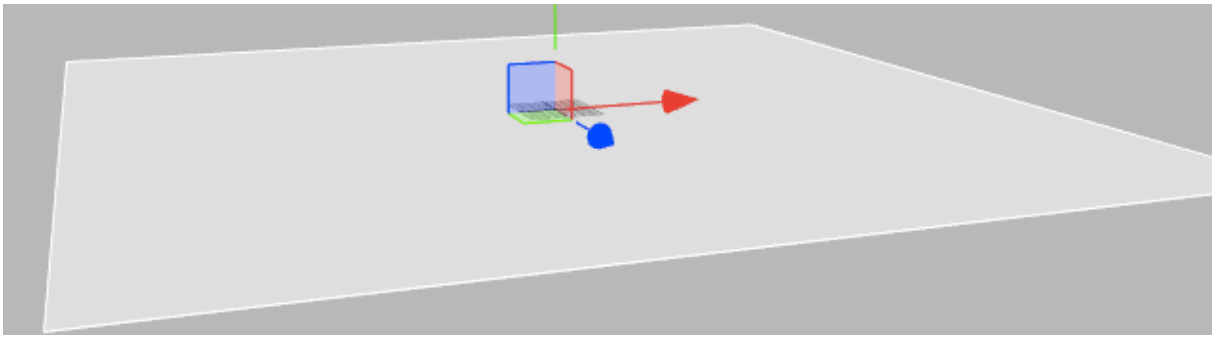
2 PlayCanvas 3D mängu näide

Mängu näitena valmistatakse esimese mängija vaatega 3D labürindi mäng. Mäng koosneb mängutegelasest, pallikujulistest objektidest, maapinnast ning seintest. Pallikujulised objektid on hajutatud mänguväljal. Mängutegelast saab juhtida WASD klahvide ning hiire abil. Hiire abil antakse suund edasi liikumiseks. Mängija ei saa liikuda läbi seinte ega maapinna. Mängu eesmärgiks on korjata kokku kõik mänguväljal olevad pallikujulised objektid.

2.1 Maapind ning seinad

Maapinna jaoks tuleb luua uus *Plane* objekt, mis sobib hästi igasuguste maapindade loomiseks. Nimetame objekti ümber *Ground* nimeliseks objektiks, et hiljem oleks nimest aru saada, mis objektiga on tegu. Maapinna suuremaks tegemiseks tuleb seda venitada ehk mõõdud tuleb muuta näite mängu puhul (100, 0, 100), mis teeb maapinna piisavalt suureks mängumaailma jaoks. Maapinnale tuleb anda ka füüsikalised omadused, et hiljem mängutegelane ei kukuks läbi maapinna. Füüsikaliste omaduste andmiseks tuleb maapinnale lisada *Rigid Body* ning *Collision* komponent. *Rigid Body* annab objektile füüsikalised omadused ja et need ka toimiks, tuleb määrata objektile Collision komponendiga füüsikaline suurus.

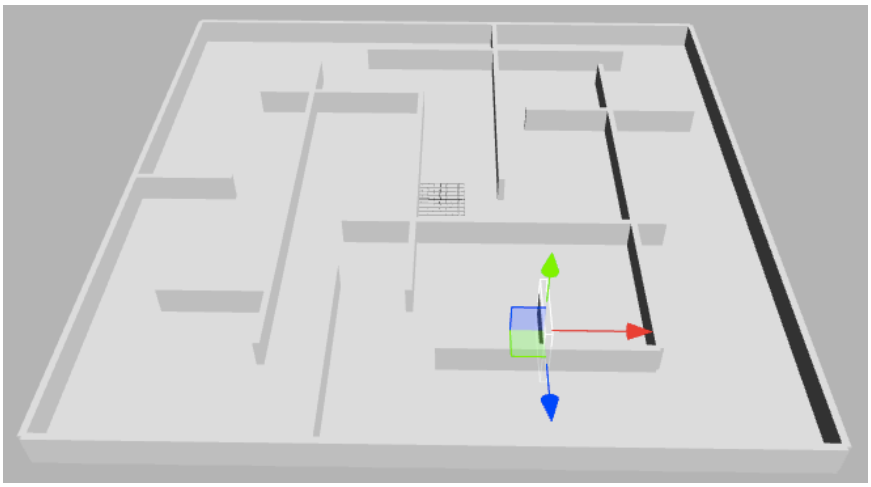
Collision komponendil tuleb määrata *Type* ehk tüüp. Tüüpideks on kast, sfäär, kapsel ja võrgusilm(*mesh*). Kast on maapinna objektile kõige sarnasem kujund. *Half Extends* määrata (50, 0, 50), mis katab terve maapinna (vt Joonis 7). *Half Extends* suurendamisel liigub selle ala sümeetriliselt mõlemalt poolt ehk terve ala täitmiseks tuleb kasutada maapinna mõõte ning need kahega jagada. *Rigid Body* puhul määrata tüüp *Static*-ks ehk staatiliseks, mis on maapinna jaoks sobilik. Sobilik sellepärast, et maapind ei ole liikuv objekt. Liikuva objekti puhul oleks see *Dynamic*.



Joonis 7 Collision valged äärejooned ümbritsevad maapinda

Seinte tegemise jaoks kasulik alguses teha uus eelseadistamata objekt ehk kaust, kuhu sisse saab paigutada kõik mängu jaoks tehtavad seinad. Eraldi kaust seinte jaoks on hea sellepärast, et muudab puu hierarhia silmale paremini loetavamaks ning üldised parameetreid on hea anda tervele kogumile. Labürindi tegemiseks on vajalikud 4 välisseina, et mängija ei liiguks mängumaailma äärelt alla ning seejärel hulk siseseinu, mis moodustavadki labürindi, mis ongi näite mängu jaoks vajalik. Välisseinte puhul on mõtekas alguses 1 sein valmis teha ning pärast seda kloonida. Välisseina puhul on oluline, et laius oleks 100, mis on maapinna laius ning kõrgus 5. Kõrguseks 5 sellepärast, et mängija ei näeks üle seinte. Peale ühe välisseina kloonimist tuleb teised seinad paigutada teistesse mängumaailma servadesse. Vastasseina puhul piisab Z või Y koordinaadi positsiooni numbri ees oleva mängi muutmisest (+ või -). Järgmise kahe seina puhul tuleb muuta ka Y rotatsioon 90ks, mis pöörab seina 90 kraadi.

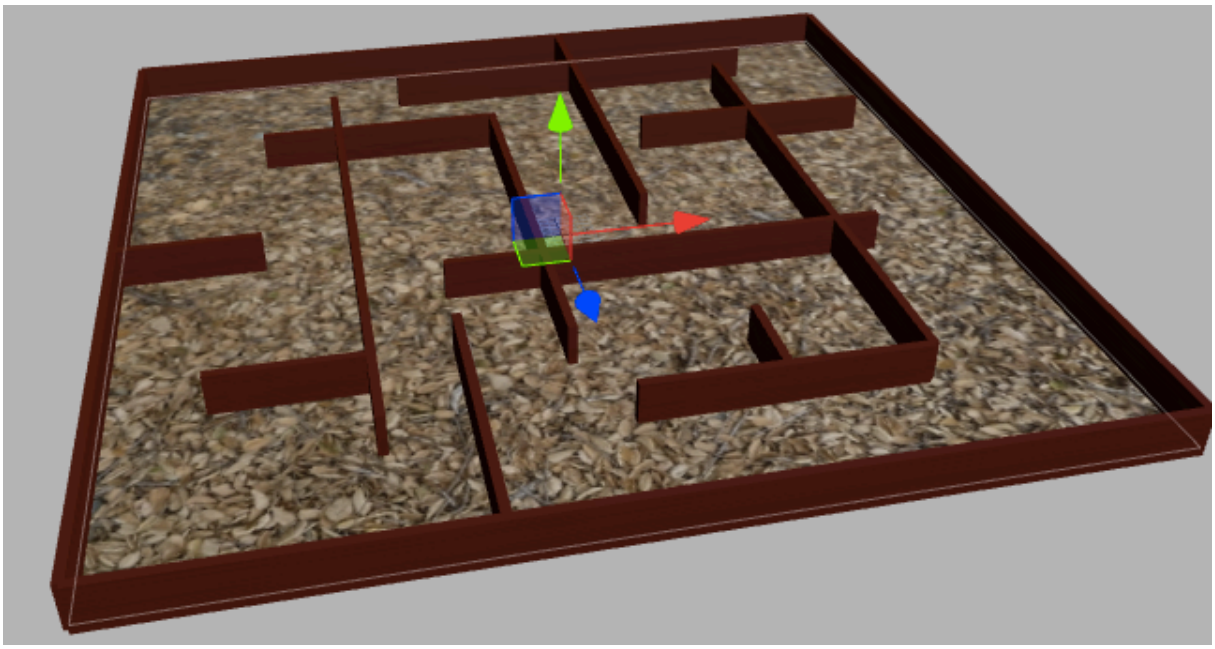
Siseseinte jaoks tuleb erineva suurusega seinu teha ning need juhuslikult paigutada. Paigutamisel arvestada seda, et mängija mahuks vahest läbi. Kahe seina vahelise käigu laiuseks minimaalselt 2. Peale vaheseinte paigutamist on vaatepilte selline(vt Joonis 8)



Joonis 8 Lõpptulemus peale seinte ja maapinna tegemist

Materjal maapinnale ja seinale

Maapinnale ja seinale ilusama vaatepildi tegemiseks tuleb neile külge panna materjal, mis tuleks ka veebiredaktori keskkonnas luua. Materjali loomiseks tuleb ressursside alt teha klikivajutus "+" nupule ning sealsest valikust valida "*New Material*". Materjali saab luua lihtsalt mõnda värvi valides või kasutada mõnda tekstuuri. Näite mängu puhul kasutatakse tekstuure, mis on saadud http://www.deviantart.com/art/Seamless-wall-texture-01-126416799 ning http://wailwulf.deviantart.com/art/Texture-Leaves-on-the-Ground-306194385 lehelt. Faili formaadilt on tekstuurid .jpg. Tekstuuride veebiredaktorisse lisamiseks tuleb need tirida salvestuskaustast ressursside alla. Peale seda saab materjalile tekstuur lisada. Tekstuuri lisamiseks materjalile tuleb materjali parameetril valida *diffuse* ning tühi materjali parameeter muuta sobivaks. Maapinnale materjali lisamiseks tuleb objekti vaatesse liikuda ning sealt "*Model*" alt valida õige materjal. Sama tegevus ka iga seina puhul. Välisseina ja siseseinte materjali puhul on sama materjal. Peale materjalide lisamist näeb mängumaailm välja selline(vt Joonis).



Joonis 9 Mängumaailm peale materjalide lisamist

2.2 Mängutegelase lisamine

Mängutegelane tuleb esimese mängija vaates ehk siis kaamera peab olema mängutegelase küljes. Esimese sammuna tuleb luua mängija, selleks tuleb teha uus tühi objekt nimega

"*Player*". Tehtud objekti sisse tuleb teha kaamera objekt, mis tekitabki esimese mängija vaate mängus.

Mängija objektile tuleb samuti lisada *Rigid Body* ning *Collision* nagu seda oli vaja nii seintel kui maapinnal. *Collisioni* puhul sobiks tüübiks *Capsule*, mille raadius 0.5 ning kõrgus 2. *Rigid Body* peab seekord olema dünaamiline erinevalt maapinnast ja seintest ehk mängija peab saama ka liikuda. Selleks, et mängija objekti liigutada saaks tuleks *Linear Damping* panna 0.99, mis annab mängijale sujuva liikumise ning *Angular* 0. *Angular* on sellepärast 0, sest mängijal pole näite mängus pöörlemist vaja. Linear Factor muuta (1, 1, 1), mis on samuti vertikaalseks liikumiseks vajalik, *Angular Factor* on (0, 0, 0). Mängutegelase liigutamiseks on vajalik skripti tegemine ning selle lisamine tegelasele, ilma selleta mängutegelane liikuma ei hakka.

Mängija liikumine

Mängija liigutamise jaoks on vaja "*Player*" objektile lisada skript(vt Lisa 1). Skripti faili tegemiseks tuleb ressursside alt teha klikivajutus "+" nupule ning sealt valida "*New Script*". Seejärel valida sellele nimi, näite mängu puhul on selleks "*Collectable*". Skriptile antakse kaasa kaamera objekt. Kaamera objekt on oluline selleks, et saada tegelasekuju edasi liikumise ja vaate suund. Mängija enese liigutamiseks on olulised klavitatuuri kuularid, et vastavalt klahvi liigutusele liiguks ka tegelaskuju õiges suunas. Skriptis tehakse liikumisele enda objekt, et seda saaks kasutada kuularite lisamiseks ning mängukaadrite uuendamiseks. Liikumise objektil on oluline tähtsus "*update()*" funktsioonil, mida käivitakse mängukaadri uuendusel. Uuendamine on oluline selleks, et igal kaadril mängija liiguks vajadusel. Uuendamise funktsiooni lisatakse ka eelnevalt mainitud klaviatuuri kuularid. Klaviatuuril kuulatakse WASD klahve, mille vajutamisel tegelaskuju liigutatakse.

2.3 Kogumisobjektid ning mänguskoor

Mänguskoor

Mänguskoori näitamiseks tuleb teha uus olem nimega "*UI*". Olemile tuleb teha uus skript "*content.js*", mille abil saaks lisada HTML ning CSS failid mängu. HTML ning CSS abil saab kuvada skoori ekraanil. Faile saab luua samamoodi nagu skripte, kuid erinevalt skripti puhul tuleb valida valikus vastavalt "*New HTML*" ning "*New CSS*". HTML nimeks on "*text.html*" ning CSS nimeks "*style.css*".

Skripti algusesse kirjutada skripti atribuudid, et saaks skripti külge panna HTML ning CSS faili.

```
pc.script.attribute("html", "asset", null, {type: "html"});
pc.script.attribute("css", "asset", null, {type: "css"});
```

Koodinäide 1 Skriptile HTML ja CSS atribuudi andmine

See koodijupp lubab nüüd veebiredaktoris content.js külge panna HTML ning CSS faili. Lisaks tuleb kirjutada initsialiseerimisse koodijupp, et see HTML ja CSS ka kuvataks mänguekraanil.

```
Content.prototype = {
        // Algseisu initsialiseerimine, html ja css
        initialize: function () {
            var asset = app.assets.get(this.html);
            if (asset) {
                var div = document.createElement("div");
                div.innerHTML = asset.resource;
                document.body.appendChild(div);
            }

            var css = app.assets.get(this.css);
            if (css) {
                style = pc.createStyle(css.resource);
                document.head.appendChild(style);
            }
        }
    };
```

Koodinäide 2 HTML ning CSS kuvamiseks

HTML faili sisuks on "Skoor:" ning algskoor "0". Algskoor ümbritseda elemendiga ning sellele anda id "score", et hiljem saaks seda JavaScript abil muuta.

```
<p>Skoor: <span id="score">0</span></p>
```

Koodinäide 3 HTML-is skoori kuvamine

Mingi stiil tuleks sellele samuti anda, et see kuvataks ekraanil mugavas kohas ning piisava suurusega, et see oleks mängijale nähtav.

```
p {
    position: absolute;
    top: 20px;
    left: 20px;
    font-size: 20px;
    color: red;
}
```

Koodinäide 4 Stiilikood skoorile

Kogumisobjektid

Kogumisobjektideks on hõljuvad pallid. Pallide tegemiseks tuleb neile teha tühi kaust, kuhu sisse palle tekitada. Pallide tekitamiseks tuleb teha uusi *sphere* objekte. Pallide Y-postsiooniks määrata 1.5, et need oleks mängijale vaatevälja keskel. Z- ja X-positsioonid määrata pallidele suvaliselt, arvestades seda, et pallid ei oleks mänguväljalt väljaspool ega seinte sees. Selleks, et mängutegelane saaks palle koguda tuleb pallidele lisada *Collision*. *Collision* tüüp on *sphere* nagu objekt ise ning raadiuseks 0.5. Mänguskoori arvestuseks on vajalik skript, kus mänguskoori üle peatakse arvet ning kogutud pallid kustutatakse mänguväljalt.

Pallide skripti tegemisel tehakse esimesena objekti initsialiseerimine ning sellele päästikud. Päästikud on olulised selleks, et saada aru, kui mõni teine objekt(näitemängu puhul mängutegelane) siseneb objekti. Skripti tuleb initsialiseerimise funktsiooni teha objekti sisse liikumise päästik.

```
Collectable.prototype = {

        // Algseisu initsialiseerimine ja päästiku tegemine

        initialize: function () {

            this.entity.collision.on("triggerenter",
this.onTriggerEnter, this);

        }};
```

Koodinäide 5 Kogumisobjekti initsialiseerimine ning päästiku tekitamine

Peale päästiku tegemist peab sellele tegema ka funktsiooni, et see ka reaalselt miskit teeks. Hetkeseisuga saab kirjutada mängutegelase sisenemisel peidetakse palli objekt, mis tuleb lisada peale initsialiseerimise funktsiooni. Lisaks saab sellele funktsioonile lisada JavaScripti jupi, mis suurendab mänguekraanil skoori 1 võrra.

```
//objekti sisenemisel lisatakse skoorile 1 punkt ning objekt peidetakse
        onTriggerEnter: function (other) {
                this.entity.model.enabled = false;
                this.entity.collision.enabled = false;
               document.getElementById('score').innerHTML     = parseInt(document.getElementById('score').innerHTML) + 1;
        }
```

Koodinäide 6 Objekti sisenemise päästiku funktsioon

Kokkuvõte

Käesoleva seminaritöö eesmärgiks oli anda WebGL'i kasutava mängumootori PlayCanvase ülevaade selle olemusest ning keskkonnas. Töö käigus näidata PlayCanvase veebiredaktori 3D mängu võimalusi näite mängu abil.

Seminaritöö tulemuseks oli 3D labürindi mäng esimese mängija vaates, mille tegemise käigus tutvustati erinevaid 3D objekte, nende omadusi ning lisasid. Mängu tegemiseks kasutati ainult PlayCanvase enda poolt pakutavat veebiredaktorit. 3D objektide materjalideks olid olemasolevad materjaid internetist - nende tegemine oleks nõudnud autoril teisi vahendeid ja lisaoskusi. Mäng on kättesaadav aadressil http://playcanv.as/b/MJatf821. Kokkuvõttes oli PlayCanvases mängu tegemine üsna hõlbus, sest tegemisel kasutasin ainult veebilehitsejat ja kuna rakendus töötab pilves, siis ei pidand mõtlema oma poolse varundamisele. Samuti oleksin lihtsalt saanud mitut arvutit kasutada mängutegemisel. Mängu tegemisel tehnoloogilisi piiranguid ette ei tulnud, kasutati veebitehnoloogiaid nagu HTML, CSS ja JavaScripti.

Seminaritöö andis autorile uusi teadmisi 3D mängu tegemisest, WebGL olemusest, PlayCanvase mängumootorist. PlayCanvase veebiredaktori kasutamine oli autori arvates üsna lihtne, mõningaid keerulisi kohti siiski oli nagu näiteks mängija liikuma saamiseks, aga need said lahendatud. PlayCanvas sobib algajamale mängutegijale, kuid ei jää nõrgaks ka juba kogenenemula oskajale.

Kasutatud kirjandus

Evans, D. (2014, 6 4). *PlayCanvas goes open source*. Retrieved 10 27, 2015, from PlayCanvas: http://blog.playcanvas.com/playcanvas-goes-open-source/

Hirshon, J. (n.d.). *Khronos Releases Final WebGL 1.0 Specification to Bring Accelerated 3D Graphics to the Web without Plug-ins*. Retrieved 10 22, 2015, from Khronos: https://www.khronos.org/news/press/khronos-releases-final-webgl-1.0-specification

Nyman, R., & Eastcott, W. (2014, 6 4). *PlayCanvas Goes Open Source*. Retrieved 10 27, 2015, from Hacks: https://hacks.mozilla.org/2014/06/playcanvas-goes-open-source/

PlayCanvas. (n.d.). *User Manual*. Retrieved 10 29, 2015, from PlayCanvas: http://developer.playcanvas.com/en/user-manual/designer/menus-and-toolbar/

PlayCanvas. (n.d.). *Plans*. Retrieved 10 29, 2015, from PlayCanvas: https://playcanvas.com/plans

Lisad

Lisa 1. Mängija liikumine

```
pc.script.attribute("camera", "entity", null);
pc.script.attribute("power", "number", 5000);
pc.script.attribute("lookSpeed", "number", 0.5);

pc.script.create('movement', function (app) {
    var force = new pc.Vec3();

    // Uue esimese mängija vaatega initsialiseerimise
    var Movement = function (entity) {
        this.entity = entity;

        this.camera = null;
        this.eulers = new pc.Vec3();
    };

    Movement.prototype = {
        //Kutsutakse välja, siis kui kõik ressursid on laetud ning enne esimest uuendamist
        initialize: function () {
            // Hiire liikumise kuular
            app.mouse.on("mousemove", this._onMouseMove, this);

            // Kursor peidetakse kui hiire nupp on all
            app.mouse.on("mousedown", function () {
                app.mouse.enablePointerLock();
            }, this);

            // Kontrollib vajalikke komponente
            if (!this.entity.collision) {
                console.error("Esimese mängija vaatega objektil on vajalik 'collision' komponent");
            }
```

```
            if (!this.entity.rigidbody || this.entity.rigidbody.type !==
pc.BODYTYPE_DYNAMIC) {
                console.error("Esimese mängija liikumise skript vajab
töötamiseks DYNAMIC 'rigidbody' komponenti");
            }
        },

        // kutsutakse igal uuel kaadril välja. dt näitab aega viimasest
uuendusest
        update: function (dt) {
            // Kontrollitakse kaamerat, kui seda pole, siis tehakse uue
            if (!this.camera) {
                this._createCamera();
            }

            // Kaamera vaatekohe saamine, et mängija liikumise suund paika
panna
            var forward = this.camera.forward;
            var right = this.camera.right;

            // liikumine
            var x = 0;
            var z = 0;

            // W-A-S-D klahvid liigutavad mängijat
            // Kontrollib klahvide vajutamist
            if (app.keyboard.isPressed(pc.KEY_A)) {
                x -= right.x;
                z -= right.z;
            }

            if (app.keyboard.isPressed(pc.KEY_D)) {
                x += right.x;
                z += right.z;
            }

            if (app.keyboard.isPressed(pc.KEY_W)) {
                x += forward.x;
```

```
                z += forward.z;
            }

            if (app.keyboard.isPressed(pc.KEY_S)) {
                x -= forward.x;
                z -= forward.z;
            }

            // suuna kasutamine kasutaja liigutamiseks
            if (x !== 0 && z !== 0) {
                force.set(x, 0, z).normalize().scale(this.power);
                this.entity.rigidbody.applyForce(force);
            }

            // kaamera nurga uuendamine
            this.camera.setLocalEulerAngles(this.eulers.y,    this.eulers.x,
0);
        },

        _onMouseMove: function (e) {
            // kui hiire pointer on lukustatud
            // kasutatakse kaamera liigutamiseks hiire liikumist
            if (pc.Mouse.isPointerLocked() || e.buttons[0]) {
                this.eulers.x -= this.lookSpeed * e.dx;
                this.eulers.y -= this.lookSpeed * e.dy;
            }
        }
    };

    return Movement;
});
```

Lisa 2. Kollekteeritava kood

```
pc.script.create('collectable', function (app) {
    // Uue Collectable tegemine
    var Collectable = function (entity) {
```

```
        this.entity = entity;
    };

    Collectable.prototype = {
        // Algseisu initsialiseerimine ja päästiku tegemine
        initialize: function () {
            this.entity.collision.on("triggerenter",    this.onTriggerEnter,
this);
        },
        //objekti  sisenemisel  lisatakse  skoorile  1  punkt  ning  objekt
peidetakse
        onTriggerEnter: function (other) {
                this.entity.model.enabled = false;
                this.entity.collision.enabled = false;
                //window.game.collectBalls(this.worth);
               document.getElementById('score').innerHTML                  =
parseInt(document.getElementById('score').innerHTML) + 1;



        }
    };
    return Collectable;
});
```

Lisa 3. UI kood

```
pc.script.attribute("html", "asset", null, {type: "html"});
pc.script.attribute("css", "asset", null, {type: "css"});

pc.script.create('content', function (app) {
    // Uue Content tegemine
    var Content = function (entity) {
        this.entity = entity;
    };

    Content.prototype = {
        // Algseisu initsialiseerimine, html ja css
        initialize: function () {
```

```
            var asset = app.assets.get(this.html);
            if (asset) {
                var div = document.createElement("div");
                div.innerHTML = asset.resource;
                document.body.appendChild(div);
            }

            var css = app.assets.get(this.css);
            if (css) {
                style = pc.createStyle(css.resource);
                document.head.appendChild(style);
            }
        }
    };
    return Content;
});
```

Lisa 4. Kujunduse kood

```
p {
    position: absolute;
    top: 20px;
    left: 20px;
    font-size: 20px;
    color: red;
}
```

Lisa 5. HTML kood

```
<p>Skoor:
      <span id="score">0</span>
</p>
```