

Jani Mutka

## **Mobiilipelin prototyypin kehitys Unity-pelimoottorilla**

## **Mobiilipelin prototyypin kehitys Unity-pelimoottorilla**

Jani Mutka  
Opinnäytetyö  
Syksy 2021  
Tietojenkäsittely  
Oulun ammattikorkeakoulu

## TIIVISTELMÄ

Oulun ammattikorkeakoulu  
Tietojenkäsittely

---

Tekijä(t): Jani Mutka

Opinnäytetyön nimi: Mobiilipelin prototyypin kehitys Unity-pelimootorilla

Työn ohjaaja(t): Matti Viitala

Työn valmistumislukukausi ja -vuosi: syksy 2021

Sivumäärä: 26 + 1 liite

---

Opinnäytetyön tarkoituksena oli tutustua yleisesti prototypointiin ja toteuttaa oman peli-idean pohjalta Android-mobiililaitteella toimiva prototyyppi. Ideana oli isometrisestä kuvakulmasta pelattava rallipeli mobiililaitteille. Pääpaino oli auton ajettavuuden ohjelmoinnissa ja hiomisessa.

Prototyyppi toteutettiin Unity-pelimootorilla ja C#-ohjelmointikielellä. 3D-malleihin ja tekstuureihin hyödynnettiin Unity Asset Storessa olevia ilmaisia resurssipaketteja. Tietolähteenä hyödynnettiin Unityn hyvää dokumentaatiota, tekijän omia ohjelmointitaitoja ja video-oppaita.

Peliprototyypin valmistuksen tavoitteina olivat oman peli-idean keskeisten elementtien testaus mobiililaitteella ja testien pohjalta tehtävä päätös, kannattaako peliä lähteä jatkokehittämään. Lisäksi tavoite oli yleisellä tasolla prototypointiin tutustuminen.

Prototyypin toteutus onnistui ja halutut ominaisuudet saatiin ohjelmoitua suunnitelman mukaisesti. Prototyypistä tehtiin asennuspaketti Android-mobiililaitteelle testausta varten. Testauksessa todettiin, että peli-idea ja toteutus toimivat mobiililaitteella ja sitä kannattaa lähteä jatkokehittämään.

---

Asiasanat: Prototyyppi, Android, Mobiilipelit, Ohjelmointi

## ABSTRACT

Oulu University of Applied Sciences  
Degree Programme in Business Information Systems

---

Author(s): Jani Mutka  
Title of thesis: Prototyping a mobile game with the Unity game engine  
Supervisor(s): Matti Viitala  
Term and year when the thesis was submitted: Autumn Term 2021  
Number of pages: 26 + 1 appendice

---

The purpose of the thesis was to get acquainted with prototyping in general and to implement a prototype that works on an Android mobile device based on one's own game idea. The idea was a rally game for mobile devices that uses an isometric camera angle. The focus was on programming car's drive controls and handling.

The prototype was implemented with the Unity game engine and the C# programming language. Free resource packs from the Unity Asset Store were utilized for 3D models and textures. Unity's good documentation, the author's own programming skills and video guides were utilized as a source of information.

The prototype was successfully implemented, and the desired features were programmed successfully. The prototype was built an installation package for an Android mobile device for testing. The testing found that the game idea and implementation works on mobile devices and is worth developing further.

---

Keywords: Prototype, Android, Mobile Games, Programming

# SISÄLLYS

1	JOHDANTO .....	6
2	PROTOTYYPPI .....	7
2.1	Historia .....	7
2.2	Erilaisia prototyypppejä .....	8
2.3	Prototypointimenetelmiä sovelluskehityksessä .....	11
3	PELIPROTOTYYPPI .....	12
4	PROTOTYYPPI OMASTA PELI-IDEASTA .....	14
4.1	Resurssit .....	14
4.2	Luonnos paperille .....	15
4.3	Auton peliobjektin rakentaminen .....	16
4.4	Ajotien rakentaminen .....	17
4.5	Auton ohjattavuuden ohjelmointi .....	19
4.6	Mobiiliohjaus .....	21
4.7	Isometrinen kamera .....	22
4.8	Ohjattavuuden hiominen .....	22
4.9	Testaus ja lopputulos .....	24
5	POHDINTA .....	26
	LÄHTEET .....	27

# 1 JOHDANTO

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa rallipelin prototyyppi Android-mobiililaitteelle oman peli-idean pohjalta. Peli-ideasta rajattiin tärkeimmät osa-alueet, ohjattavuus, ajettavuus ja isometrinen kamerakulma, joiden toimivuus mobiililaitteella haluttiin testata. Lisäksi tavoitteena oli tutustua yleisesti eri alojen prototyypin menetelmiin ja historiaan. Oma peliprototyyppi toteutettiin Unity-pelimootorilla ja ohjelmointikielenä oli C#. Prototyyppi tehtiin itsenäisesti, mutta projektissa hyödynnettiin ilmaisia resursseja 3D-mallien ja tekstuurien osalta.

Opinnäytetyössä käydään läpi lyhyesti prototypoinnin historiaa, prototyypin eri muotoja teollisuudessa ja pelikehityksessä sekä oman prototyypin toteuttaminen. Oma peli-idea esitellään ja prototyypin toteutusta käsitellään valmistusjärjestyksessä ja keskeisiä elementtejä, kuten komponenttivalintoja ja skriptejä avataan tarkemmin.

Oman peliprototyypin toteutuksessa hyödynnettiin omaa ohjelmointi- ja Unity-pelimootorituntemusta, Unityn dokumentaatiota, ilmaisia Unity Asset Storen resursseja, C# dokumentaatiota sekä opetusvideoita ja ohjeita. Tavoitteena oli tehdä Android-mobiililaitteella toimiva peliprototyyppi, joka sisältää aiemmin mainitut tärkeimmät ominaisuudet. Prototyyppiä testaamalla oli tarkoitus varmistaa, soveltuuko peli mobiilialustalle ja onko sitä kannattavaa lähteä jatkokehittämään.

## 2 PROTOTYYPPI

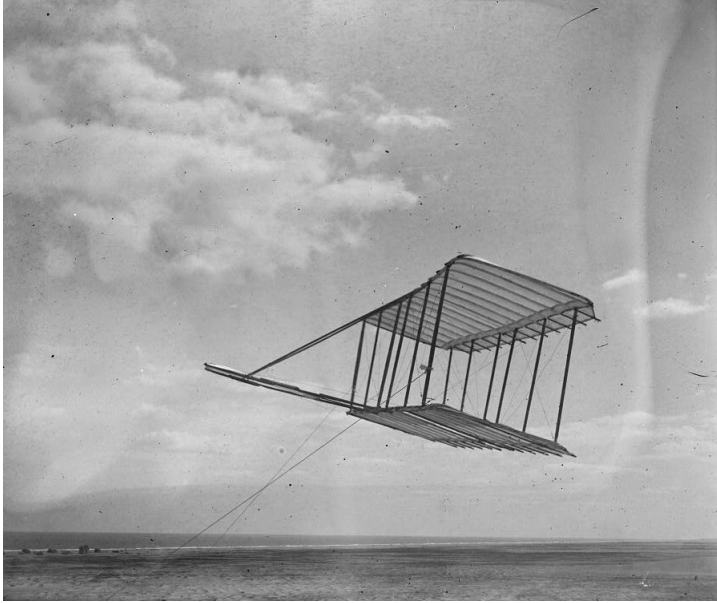
Prototyyppi on varhainen näyte, malli tai tuote, joka on luotu konseptin, idean tai prosessin testaamiseksi. Se on tärkeä osa suunnitteluprosessia ja käytäntö, jota käytetään kaikilla suunnittelualoilla. Arkkitehdit, insinöörit, teolliset suunnittelijat ja palvelusuunnittelijat tekevät prototyyppijä testataksseen suunnitteluaan ennen massatuotantoon investointia. (Creately 2021.)

Suuri osa prototyypeistä epäonnistuu testausvaiheessa, ja näin ollen suunnitteluvirheet paljastuvat. Tämän ansiosta vältetään turhan ajan ja rahan tuhlausta heikkojen tai sopimattomien ratkaisujen toteuttamisessa. Prototyyppien etuna on se, että koska investointi on pieni, riski on pieni. (Creately 2021.)

### 2.1 Historia

Ajatus prototyyppien tekemisestä juontaa juurensa vuosisatoja ihmiskunnan historiaan, jolloin keksijät loivat toimivia mittakaavamalleja tutkimisen ja testaamisen helpottamiseksi. Leonardo Da Vinci on nykyään lähes yhtä kuuluisa keksinnöistään kuin taiteestaan. Da Vinci tutki monia ideoita, jotka ovat nykyään todellisuutta, kuten lentokoneita, laskuvarjoja ja jopa robotteja. (Bennett 2020.)

1900-luvun alussa Wrightin veljekset suunnittelivat ja valmistivat ensimmäisen lentokoneen. He aloittivat työn tutkimalla aerodynamiikkaa ja etsivät vastauksia lennon ongelmiin tarkkailemalla isojen lintujen liitoa. He testasivat varhaisia leija- ja purjelentokoneprototyyppijä yli kahdella sadalla erilaisella siivellä ja kantopintamallilla (kuvio 1.). Muita kehityksen aikana testattuja prototyyppijä olivat muun muassa moottorit, potkurit ja lennonohjausjärjestelmät. Kaikki tämä johti ensimmäiseen onnistuneeseen lentoon moottoroidulla ja ohjattavalla lentokoneella vuonna 1903. Prosessi, jota Wrightin veljekset käyttivät ensimmäisen lentokoneen kehittämiseen, on täysin sama, mitä NASA käyttää vielä nykyäänkin ongelmien ratkaisemiseen. (NASA 2021.)



KUVIO 1. Purjelentokoneen lento- ja liito-ominaisuuksien testaus (Wright-brothers.org 2011)

1960-luvulle asti ihmiset tekivät prototyyppejä manuaalisesti eri mittasuhteissa tai täysin toimivilla malleilla, mutta tietokoneiden keksiminen muutti myös prototypointia. 1960-luvulla insinööriprofessori Herbert Voeckler tutki mahdollisuuksia käyttää tietokoneita tehtaan työstökoneiden ohjaamiseen. Tämän tuloksena Voeckler kehitti ensimmäiset matemaattiset teoriat ja algoritmit, jotka muodostivat perustan nykyisille 3D-sovelluksille. 3D-ohjelmistolla on sittemmin suunniteltu kaikkea mekaanista leluautoista pilvenpiirtäjiin. (Bennett 2020.)

Möhemmin, 1980-luvun lopulla, Carl Deckard oli edelläkävijä kerrospohjaisessa valmistuksessa. Deckard kehitti SLS-tekniikan (Selective Laser Sintering), joka mahdollisti muovikappaleiden tulostamisen. Voeckler ja Deckard olivat nopean protypoinnin edelläkävijöitä ja mullistivat valmistusprosessin. Nyt 3D-ohjelmistot ja 3D-tulostus antavat meille mahdollisuuden prototyyppien nopean valmistuksen lisäksi myös tuotteiden valmistuksen omassa kodissamme. (Bennett 2020.)

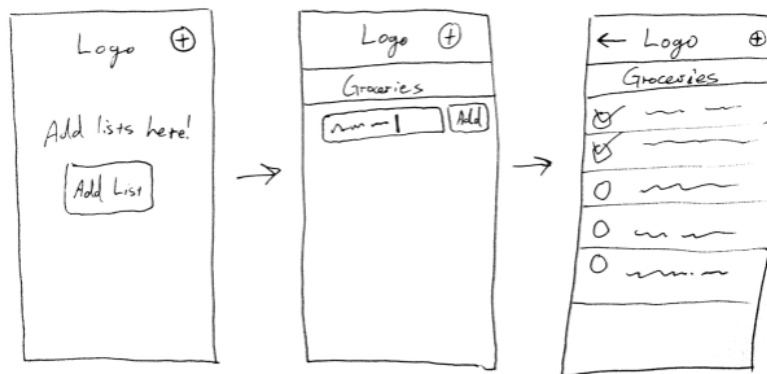
## 2.2 Erilaisia prototyyppejä

Projektien eri vaiheissa hyödynnetään erilaisia prototyyppimenetelmiä. Tässä listattuna erilaisia prototyyppityylejä, joita hyödynnetään fyysisten ja digitaalisten tuotteiden kehityksessä.

Piirtäminen on tehokas tapa havainnollistaa ideat ja konseptit visuaaliseen muotoon (kuvio 2.) ja sitä käytetään lähes aina tuotteen suunnittelussa. Tämä on ehkä alkeellisin prototyyppimuoto,



mutta piirretyn luonnoksen käyttäminen on edelleen laajalti käytetty tapa jakaa konsepti. (Cui 2018.)



KUVIO 2. Piirretty luonnos mobiilisovelluksesta (Treehouse 2014)

Rautalankamalli on staattinen esitys eri näytöistä ja sivuista, joista tuote muodostuu (kuvio 3.). Rautalankamallissa käytetään yksinkertaisia muotoja luomaan visuaalisia esityksiä esimerkiksi verkkosivun asettelusta. Niitä käytetään viestimään yksittäisten sivujen rakenteesta (miten sivun osat toimivat yhdessä ja missä sisältö tulee olemaan) ja miten sivut yhdistetään (miten käyttöliittymä toimii käyttäjän näkökulmasta). (Babich 2021.)



KUVIO 3. Rautalankamalli Adobe XD-ohjelmassa (Babich 2020)

Prototyyppimalli voidaan valmistaa mistä tahansa materiaalista aina rakennuspalikoista käsityöpaperiin (kuvio 4.). Tarkoituksena on antaa karkea käsitys suunnittelusta, eikä siinä tarvitse olla toimivia osia. Tämä prototyyppi toimii hyvin skaalatuissa konsepteissa ennen suuren mallin

tuottamista. Fyysisten mallien prototyypit voivat olla hyödyllisiä pienemmille esineille tai jopa arkkitehtuurisille suunnitelmille. (Indeed 2021.)



*KUVIO 4. Kääntyvän liiketunnistinalustan prototyyppi*

3D-tulostus on mahdollistanut nopean prototypoinnin (Rapid prototyping). Nopean prototypoinnin avulla voidaan siirtyä suunnittelusta tuotantovaiheeseen nopeammin, koska tulostettua 3D-mallia voidaan käyttää virheiden tai ongelmien tunnistamiseen (kuvio 5.). Kun suunnittelu on valmis, sitä voidaan helposti muokata havaintojen ja testien perusteella. Prototyypitiedostoon on helppo tehdä tarvittavia muutoksia. 3D-tulostuksen nopeus tuo merkittäviä säästöjä kehitykseen, ja sitä hyödynnetään lähes kaikkialla teollisuudessa, esimerkiksi yksinkertaistamaan suuria malleja, kuten moottoreita ja lentokoneiden osia. 3D-tulostuksen avulla voidaan toteuttaa myös toimivia prototyyppimalleja. Toimiva prototyyppimalli mahdollistaa idean testauksen, jotta nähdään, toimiiko se halutulla tavalla. Tästä on hyötyä koneistetuille tuotteille tai muille malleille, joilla on ominaisuuksia, joiden on liikutettava tai sovittava tietyllä tavalla. (Velling 2020.)



*KUVIO 5. 3D-tulostettuja hiiren prototyyppejä (Zmorph SA 2016)*

Lisätty todellisuus- ja virtuaalitodellisuustekniikkojen (Augmented reality, Virtual reality) kehitys on mahdollistanut prototyyppien luomisen virtuaaliseen ympäristöön. Tekniikkaa voidaan hyödyntää suurien kokonaisuuksien, kuten talon tai vaikka ostoskeskuksen suunnittelun tarkasteluun virtuaaliympäristössä. (Sampath 2018.)

## **2.3 Prototypointimenetelmiä sovelluskehityksessä**

Ohjelmistokehityksessä prototyyppiä rakennetaan, testataan ja muokataan, kunnes hyväksyttävä prototyyppi on saavutettu. Se myös luo pohjan lopullisen järjestelmän tai ohjelmiston tuottamiseksi. (Martin 2021.)

Rapid Throwaway -prototyyppi perustuu alustaviin vaatimuksiin. Se kehitetään nopeasti ja tarkoituksena on näyttää, miltä vaatimukset näyttävät visuaalisesti. Asiakkaan palaute auttaa vaatimusten päivittämisessä, ja prototyyppi luodaan uudelleen, kunnes vaatimus on perusteltu. Tässä menetelmässä kehitetty prototyyppi hylätään, eikä se ole osa lopulta hyväksyttyä prototyyppiä. Tämä tekniikka on hyödyllinen ideoiden tutkimiseen ja välittömän palautteen saamiseen asiakkaiden tarpeista. (Martin 2021.)

Evoluutioprototypointi (Evolutionary Prototyping) on hyödyllinen projektille, jossa käytetään uutta tekniikkaa, jota ei vielä ymmärretä hyvin. Sitä käytetään myös monimutkaisissa projekteissa, joissa jokainen toiminnallisuus on tarkistettava kerran. Siitä on apua, kun vaatimus ei ole vakaa tai sitä ei ymmärretä selvästi alkuvaiheessa. Kehitettyä prototyyppiä jalostetaan asteittain asiakkaan palautteen perusteella, kunnes se lopulta hyväksytään. Se auttaa säästämään aikaa ja vaivaa. (Martin 2021.)

Inkrementaalisisessa prototypoinnissa (Incremental Prototyping) lopputuote pilkotaan erillisiksi pieniksi prototyypeiksi ja kehitetään yksilöllisesti. Lopulta eri prototyypit yhdistetään yhdeksi tuotteeksi. Tämä menetelmä auttaa vähentämään käyttäjän ja sovelluskehitystiimin välistä palauteaikaa. (Martin 2021.)

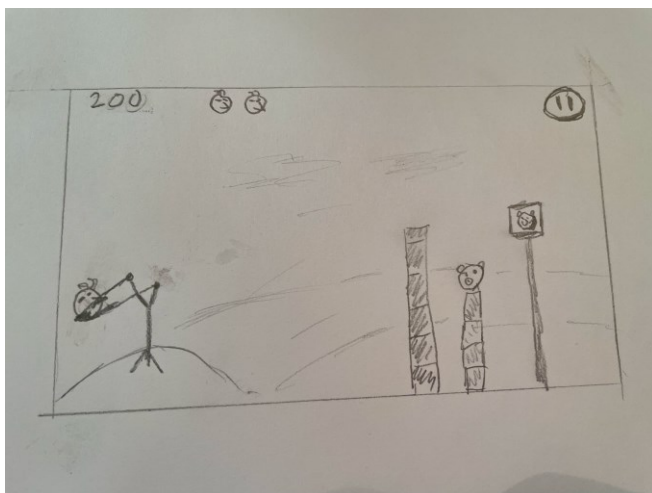
### 3 PELIPROTOTYYPPI

Prototyypin tekeminen on tärkeä osa myös videopelin kehitystä ja hyvä tapa arvioida pelikonseptin pätevyys ja saada tarpeeksi tietoa hyödyllisen pelisuunnitteludokumentin kirjoittamiseen. On suositeltavaa tehdä fyysinen, ja jos mahdollista, raaka ohjelmallinen prototyyppi peli-ideasta ennen lopullisen pelisuunnitteludokumentin tekemistä. (Fullerton 2014.)

Prototyyppiä ei pidä sekoittaa pelin alpha-versioon. Prototyypin avulla voidaan vahvistaa, että peli-ideaan kannattaa sijoittaa kuukausien tai jopa vuosien työpanos. Se sisältää vain ydinpelimekaniikan, joka todella määrittelee pelin. Grafiikan ei tarvitse olla hienoa, eikä erikoisefektejä tarvita, vaan tärkeintä on, että ydinpelimekaniikka tulee esille. (Victorino 2015.)

Videopelin valmistuksessa on monia eri vaiheita. Projektin kehityskaaren aikana hyödynnetään myös monenlaisia prototyyppimenetelmiä, joista listaan yleisimmin käytetyt.

Peli-idean ensimmäinen visuaalinen muoto on helppo toteuttaa piirtämällä pelinäkymä paperille (kuvio 6.). Piirtotaidolla ei ole väliä, vaan tärkeää on saada keskeinen osa ideaa paperille. Se voi olla pelihahmo, maisema, pelitilanne tai muu keskeinen osa pelitilannetta. Varsinkin mobiilipelien kohdalla käytetään pelitilannepiirrosta, jossa matkapuhelimen rajalliseen ruudunkokoon piirretään pelinäkymä ja tarvittavat näytölle sijoitettavat käyttöliittymäobjektit.



KUVIO 6. Esimerkkiluonnos

Konseptitaide on tarkempi visuaalinen kuvaus hahmosta (kuvio 7.), paikasta tai esineestä, jota voidaan käyttää tulevassa pelissä. Piirtämällä hahmotellaan ja luodaan ensimmäinen katsaus siihen, miltä idea voisi näyttää pelissä; esimerkiksi idea pelihahmosta yritetään visualisoida eri tavoilla luomalla useita viimeisteltyjä ja yksityiskohtaisia malleja, jotta löydetään paras yleisilme hahmolle. (Fitzgerald 2021.)



KUVIO 7. Pelihahmon konseptitaidetta (4art Media 2021)

Pelille keskeisen pelimekaniikan testaaminen on erittäin tärkeää ja paras tapa tähän on prototyypin tekeminen pelimoottorilla. Kehittyneet pelimoottorit, kuten Unity, mahdollistavat pelattavan prototyypin tekemisen hyvin nopeasti. Pelimoottorit sisältävät valmiita työkaluja ja resurssipaketteja, joiden avulla prototyypin voi rakentaa ilman, että aikaa kuluu prototyypin kannalta epäolennaiseen työhön.

Pelattavassa prototyypissä grafiikan ei tarvitse olla hienoa – karkeat mallit keskeisille objekteille riittävät. Myöskään koodin ei tarvitse olla puhdasta, vaan riittää, että tärkeät toiminnot ovat toimivia (kuvio 8.). Prototyypin koodia ei yleensä hyödynnetä suoraan valmiissa pelissä. (Grassi 2021.)



KUVIO 8. Kuva Squadron-pelin prototyypistä (DeerStudios 2021)

## 4 PROTOTYYPPI OMASTA PELI-IDEASTA

Olen suunnitellut mobiilialustalle rallipelin, jossa voidaan ajaa yksittäisiä erikoiskokeita tai kokonaisia ralleja. Kisat ajetaan tekoälykuskien aikoja vastaan. Kisat sijoittuvat eri maihin ja ajo-olosuhteet sekä teiden pinnat vaihtelevat maiden välillä. Tarkempi kuvaus peli-ideasta löytyy ajatuskartasta (liite 1.). Ajettavuudeltaan pelin on tarkoitus olla hauska, mutta myös sopivan haastava, jotta mielenkiinto aikojen parantamiselle säilyy. Tämän idean pohjalta toteutettiin pelattava prototyyppi. Tärkeimpänä asiana pidin ajettavuutta, ohjaustuntumaa ja mobiililaitteella pelattavuutta, joten ensimmäinen prototyyppi rakennettiin näiden osa-alueiden testaukseen.

### 4.1 Resurssit

Pelattavan prototyypin luomiseen käytettiin Unity-pelimootoria ja sen mukana tulevia työkaluja. Lisäksi hyödynnettiin valmiita 3D-malleja, jotta kehityksessä pystyttiin keskittymään tehokkaasti prototyypin tärkeimpiin osa-alueisiin: ohjaustuntumaan, ajettavuuteen ja mobiililaittepelattavuuteen. Seuraavaksi esitellään prototyypin tekemiseen hyödynnettyjä resursseja.

Unity on Unity Technologiesin kehittämä pelimootori, joka julkistettiin kesäkuussa 2005. Sillä voidaan luoda kolmiulotteisia (3D) ja kaksiulotteisia (2D) pelejä sekä interaktiivisia simulaatioita ja muita kokemuksia. Mootori tukee useita työpöytä-, mobiili-, konsoli- ja virtuaalitodellisuusalustoja. Laajan alustatuen ja monipuolisen työkalutarjonnan ansiosta se on suosittu sekä harrastekehittäjien että pelistudioiden keskuudessa. Sillä voidaan hallita muun muassa 2D- ja 3D-fysiikoita, peliobjektien liikettä, grafiikan renderöintiä ja valaistusta. Unityä käytetään myös videopelien ulkopuolisilla teollisuudenaloilla, kuten elokuva- ja autoteollisuudessa, arkkitehtuurissa, suunnittelussa ja rakentamisessa. Unityn perusversio on maksuton. (Petty 2021: Unity Technologies 2021a.)

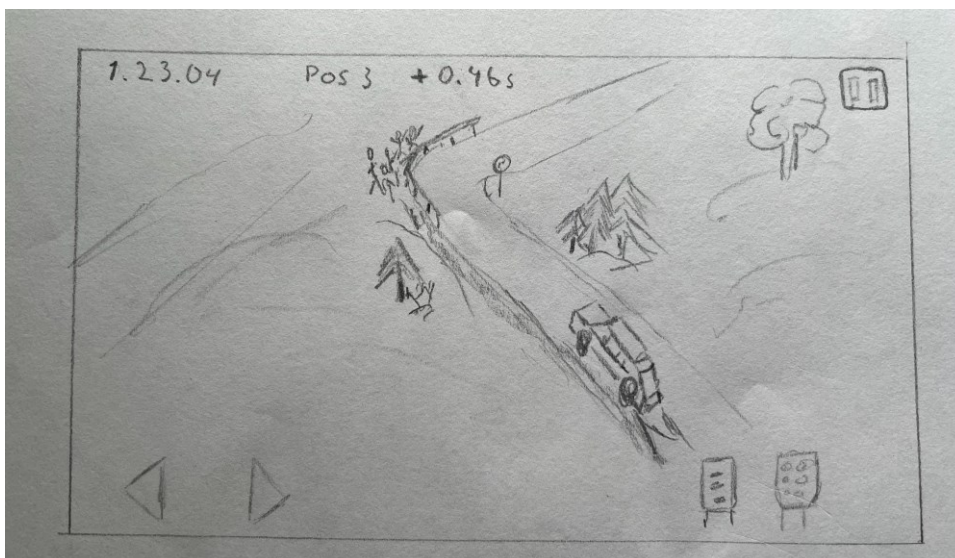
Unity Asset Store on Unity Technologies -resurssikirjasto. Siellä on erilaisia Unity Technologiesin ja käyttäjien luomia resursseja, kuten tekstuureja, animaatioita ja 3D-malleja sekä kokonaisia projektiesimerkkejä, opetusohjelmia ja editorilaajennuksia. Saatavilla on ilmaisia ja kaupallisia resursseja, jotka voidaan ladata suoraan Unity-projektiin. (Unity Technologies 2021b.)

Microsoft Visual Studio on Microsoftin valmistama IDE (ohjelmointiympäristö), jota käytetään erityyppiseen ohjelmistokehitykseen, kuten tietokoneohjelmiin, verkkosivustoihin, verkkosovelluksiin, verkkopalveluihin ja mobiilisovelluksiin. Se sisältää täydennystyökaluja, kääntäjiä ja muita ohjelmistokehitysprosessia helpottavia ominaisuuksia. Unity-pelimoottorissa voidaan hyödyntää Visual Studio -integraatiota ja C#-ohjelmointikieltä. (Incredibuild 2021: Unity Technologies 2021c.)

Unity Remote on puhelimeen asennettava sovellus, jonka avulla peliä voidaan testata suoraan kohdelaitteessa ilman, että projektia tarvitsee kääntää, siirtää ja asentaa testipuhelimeen jokaista testausta varten. Puhelin yhdistetään tietokoneeseen USB-kaapelilla, ajo näkyy laitteen näytöllä ja puhelimen komennot lähetetään takaisin käynnissä olevaan projektiin Unityssa. Sovellus on saatavilla sekä Androidille että iOS:lle. (Unity Technologies 2021d.)

#### 4.2 Luonnos paperille

Visuaalisesti valmiin pelin on tarkoitus olla yläviistosta kiinteällä kuvakulmalla pelattava. Auto sijoituu kameraan niin, että suurin osa tyhjistä tilasta on auton edessä. Ajon aikana ruudun ylälaidassa näkyy erikoiskokeen ajanotto, väliaika ja sijointi. Napit kaasulle, jarrulle ja ohjaukselle sijaitsevat ruudun alareunassa ja valikko-keskeytys-näppäin oikeassa ylälaidassa. Prototyypin ensimmäisessä vaiheessa näiden ajatusten pohjalta piirrettiin luonnos pelinäköymästä (kuvio 9.).



KUVIO 9. Luonnos pelinäköymästä

### 4.3 Auton peliobjektin rakentaminen

Kun ideointi ja pelitilanteen visuaalinen hahmottelu oli valmis, aloitettiin prototyypin rakentaminen Unity-pelimootorilla. Auton 3D-mallina hyödynnettiin maksutonta Unityn Asset Storesta ladattavaa Car'Toon: The Sport Car with interior -autoa (Unity Technologies 2021e). 3D-mallissa auto rakennetaan useasta itsenäisestä 3D-mallista (kuvio 10.). Tästä on hyötyä esimerkiksi silloin, jos autoon halutaan tehdä vauriomallinnusta. Renkaan puhkeamisen visualisointi voidaan toteuttaa korvaamalla kolarihetkellä ehjän renkaan 3D-malli rikkiäisen renkaan 3D-mallilla.



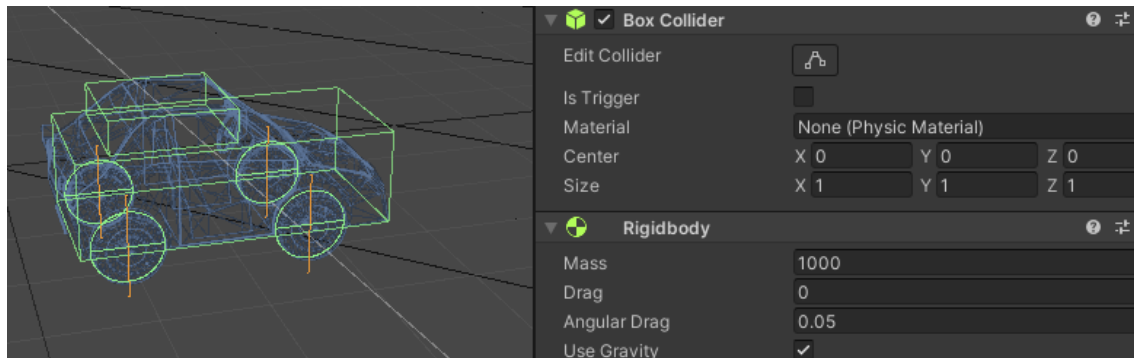
KUVIO 10. Auton 3D-mallin elementit

Unityn fysiikkamootorin hyödyntäminen tapahtuu RigidBody-komponentin kautta. Heti RigidBody-komponentin lisäämisen jälkeen auton runkoon vaikuttaa painovoima ja muun muassa auton massa on säädettävissä komponentin Mass-parametrilla (Unity Technologies 2021f). Aluksi autolle annettiin massa-arvoksi 1000 yksikköä, mutta Unityn käyttöliittymän ansiosta arvon hienosäätö testivaiheessa sopivan massan saamiseksi on helppoa.

Peliobjektien törmäysten tunnistuksesta vastaa Collider-komponentti, joita on valmiiksi eri muotoisia ja niiden koko on täysin muokattavissa (Unity Technologies 2021g). Prototyypin autoon riittävän tarkka törmäystunnistus saatiin tehtyä lisäämällä ala- ja yläosiin suorakaiteen muotoiset Box Collider -komponentit ja säätämällä ne auton kokoon sopiviksi (kuvio 11.). Törmäyslaatikot eivät muokailu täysin auton 3D-mallia, mutta se ei ole ongelma varsinkaan prototyyppivaiheessa. Unityllä on mahdollista luoda hyvinkin tarkkoja törmäystunnistusmuotoja, mutta mitä monimutkaisempi



muoto on, sitä raskaammaksi törmäystunnistuksen laskeminen käy prosessorille, joten varsinkin mobiilialustalle suunnatun pelin kehityksessä tämä on hyvä huomioida.



KUVIO 11. Komponenttien lisääminen 3D-malliin

Auton 3D-mallin renkaissa hyödynnettiin Wheel Collider -komponenttia (Unity Technologies 2021h), joka on perinteistä Collideria monipuolisempi ja suunniteltu erityisesti renkailla varustettujen ajoneuvoille. Sillä pystytään törmäystunnistuksen lisäksi vaikuttamaan myös rengasmaisiin ominaisuuksiin, kuten liike- ja sivusuunnan pitoon sekä jousitukseen. Jokaiselle renkaalle lisättiin oma Wheel Collider ja aluksi käytettiin parametrien oletusarvoja, joita muokattiin ohjattavuuden ohjelmoinnin ja testauksen yhteydessä oikean ajotuntuman saavuttamiseksi.

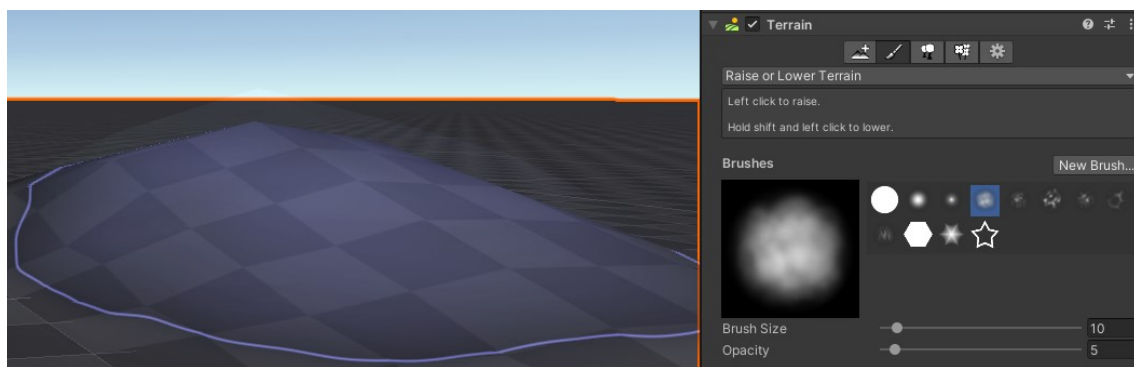
#### 4.4 Ajotien rakentaminen

Auton ajettavuuden tarkempaa testausta varten rakennettiin testirata, johon lisättiin pieniä hyppyjä, epätasaisuuksia ja esteitä. Unity Asset Store -kaupassa on tarjolla lukuisia maksullisia ja maksutomia työkaluja, jotka helpottavat erilaisten teiden ja polkujen tekoa, mutta auton ajo-ominaisuuksien prototypointiin ja mobiililaitetestaukseen nämä eivät olisi tuoneet lisäarvoa. Jokaisessa Unityyn ladattavassa lisätyökalussa on omanlaisensa käyttöliittymä sekä toiminnallisuudet, joihin tutustuminen ei palvellut prototyyppivaiheen tarkoitusta. Tehokkain tapa oli hyödyntää jo ennestään tuttua Unityn sisäänrakennettua Terrain-työkalua. Työkalulla voidaan luoda isojaakin pelialueita ja pinnan muotoja voidaan nostaa ja laskea helposti hiiren klikkauksilla. Pelialueeseen voidaan myös lisätä tekstuureja, ruohoa ja puita nopeasti. (Unity Technologies 2021i.)

Terrain-työkalulla maa-alueita lisätään yksi kerrallaan, ja oletuksena yksi alue on Unityn oletusmitakaavassa yhden neliökilometrin kokoinen, joten yksi alue oli riittävän iso testiradan rakentamiseen. Oletuksena lisätty alue on tasainen ja harmaa skriptiohjattu taso, jota voidaan muokata

pensselityökalulla. Samalla työkalulla tasoon voidaan tehdä korkeuseroja, lisätä 3D-objekteja ja tekstuureja.

Ensimmäisenä luotuun tasoon tehtiin maanmuotoja ja ajotietä varten tienpinnan korkeuseroja, jotta ajettavuutta pystyttiin testaamaan todellisemmassa ajoympäristössä. Pensselityökalussa voitiin valita pensselin vaikuttavan alueen koko ja tehokkuus. Lisäksi pensselille oli mahdollista valita eri muotoisia päitä, joita vaihtelemalla muodoista saatiin luonnollisempia (kuvio 12.). Kun mäet ja muodot saatiin tehtyä, tasoitettiin muotojen läpi ajotie kevyemmällä asetuksilla, jolloin maan muotoja säilyi tasossa, mutta tie oli selkeästi erotettavissa muista muodoista.



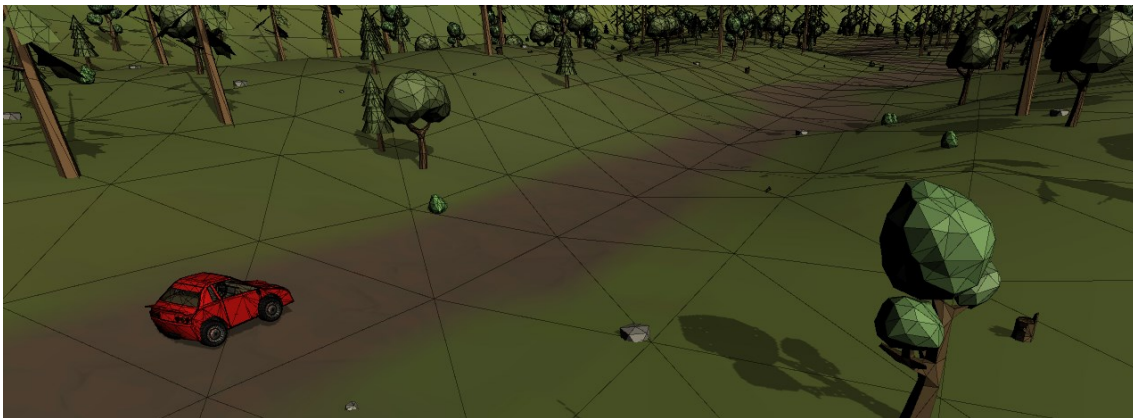
KUVIO 12. Pelialueen luonti Terrain-työkalulla

Jotta auton liikkeen ja nopeuden pystyi havaitsemaan paremmin, päädyttiin pelialueelle lisäämään myös tekstuurit ja staattisia peliobjekteja, kuten kiviä ja puita. Tekstuureina hyödynnettiin Asset Storesta löytyvää maksutonta FREE Stylized PBR Textures Pack -maatekstuuripakettia (Unity Technologies 2021j) ja peliobjekteina Low-Poly Simple Nature Pack -pakettia, joka löytyi myös Asset Storesta (Unity Technologies 2021k).

Terrain-työkalussa tekstuurit tiilittyvät ja tiilien kokoa ja tarkkuutta voidaan säätää tarkoitukseen ja tekstuurille sopivaksi. Koko tasolle voidaan valita pohjatekstuuri, jonka jälkeen tasolle voidaan lisätä eri tekstuureja pensselityökalun avulla. Valittavissa ovat samat säädöt kuin maan muotojen tekemisessä ja hyödynnettävissä on myös erilaisia pensselimuotoja. Tietekstuuri maalattiin aiemmin muovatun ajotien päälle ja tekstuurien tarkkuutta ja tiilikokoa säädettiin, jotta tekstuuriitiilien toisto ei ole pelatessa häiritsevää.

Kun ajotie oli valmis, lisättiin pelialueelle myös staattiset objektit. Objekteja voidaan sijoittaa pelialueelle manuaalisesti yksi kerrallaan, mikä pelialueen koon huomioon ottaen ei ollut järkevä

vaihtoehto. Parempia ja nopeampia vaihtoehtoja olivat Terrain-työkaluun sisältyvät pensselityökalulla piirtäminen tai massasijoitus. Piirtäminen tapahtuu hyvin samalla tavalla, kuin tekstuurienkin lisääminen; valitaan peliobjekti, esimerkiksi puu, ja määritellään sille koon vaihteluväli, ilmestymistiheys ja siveltimen koko. Tämän jälkeen maalataan pelialuetta, jolloin puita ilmestyy asetusten mukaisesti tasolle. Tätä vaihtoehtoa kokeiltiin aluksi, mutta nopeammin ja huomattavasti luonnollisempi lopputulos saatiin aikaan massasijoituksella. Massasijoitustyökaluun valitaan kaikki objektit, joita halutaan lisätä tasolle. Koko ja vaihteluvälimääritykset tehdään samalla tavalla, kuin pensselillä maalatessa. Tämän jälkeen työkalu lisää valittuja objekteja koko pelialueelle yhden napin painalluksella. Kun objektit oli lisätty, käytiin ajotie läpi ja poistettiin tielle ilmestyneet objektit. Näillä työkaluilla saatiin nopeasti prototyyppiin soveltuva pelialue, jossa ajettavuutta oli mahdollista testata (kuvio 13.).



KUVIO 13. Pelialue, johon on lisätty tekstuurit ja staattisia objekteja

#### 4.5 Auton ohjattavuuden ohjelmointi

Auton liikuttamista ja ohjausta varten tarvitaan skripti, jonka toteutukseen oli monia tapoja. Aluksi testattiin tapaa, jossa lisätään liike-energiaa koko auto-peliobjektille ja ohjauksessa eturenkaita käännetään haluttu astemäärä. Tämä lähestymistapa osoittautui täysin toimivaksi ratkaisuksi, mutta samalla osa renkaissa käytettävän Wheel Collider -komponentin ominaisuuksista jäisi hyödyntämättä. Tällä toteutustavalla auton kiihdyttäminen vaikuttaa koko auto-peliobjektiin, ja renkaat vain pyörivät liikkeen mukana, mikä poistaa sutimiskitkan vaikutuksen ajettavuudesta.

Wheel Collider -komponentti sisältää ominaisuudet motorTorque (moottorivääntö), jolla rengasta voidaan pyörittää eteen ja taakse, sekä breakTorque (jarruvääntö), jolla pyörimisnopeutta voidaan hidastaa. Näitä ominaisuuksia testattiin ajoskriptissa, ja lopputuloksena oli aidompi sekä

monipuolisempi ajomalli, joka mahdollisti auton ajo-ominaisuuksien tarkemman muokkauksen verrattuna ensimmäisenä kokeiltuun tapaan. Muutoksen ansiosta kiihdytyksessä on mahdollista, että renkaat sutivat ja renkaan kitkaominaisuudet tulevat muutenkin paremmin esille. Lisäksi tien pintamateriaalin vaikutus renkaan pito-ominaisuuksiin on helpompi lisätä jatkokehitysvaiheessa. Kuvi-  
ossa 14 kerrotaan tarkemmin ajoskriptin keskeinen funktio, jolla auton ohjaus ja liikkuminen tapahtuu.

```
// Funktio auton liikuttamiseen, jota kutsutaan joka kuvapäivityksellä suoritettavasta
// update-funktiosta.
// Parametreinä vastaanotetaan kiihdytys, ohjaus, ja jarrutus.
2 references
void Go(float accel, float steer, float brake)
{
    // Jos vastaanotettu parametri on määriteltyjen rajojen sisällä määritellään
    // se sen arvo muuttujaan esim. jos steer parametrinä vastaanotetaan 1,
    // kerrotaan se ohjauskulmalla ja tämä arvo tallentuu
    // steer muuttujaan jota päivitetään päivitysmuuttujassa eturenkaille.
    accel = Mathf.Clamp(accel, -1, 1);
    steer = Mathf.Clamp(steer, -1, 1) * steeringAngle;
    brake = Mathf.Clamp(brake, 0, 1) * maxBrakeTorque;
    // Väännölle annetaan haluttu voima joko eteen tai taakse(- tai +) riippuen
    // accel-muuttujan arvosta.
    float thrustTorque = accel * torque;

    // for-loopissa käydään wheel collider kerrallaan läpi kullekin
    // renkaalle päivitettävä arvo.
    // Jokaiselle renkaalle on oma wheel collider, jotka ovat WCs taulukossa (array).
    // Taulu käydään läpi ja muuttujan arvo lisätään kunkin renkaan wheel colliderille
    // riippuen onko kyseessä etu vai taka rengas
    for(int i = 0; i < 4; i++)
    {
        WCs[i].motorTorque = thrustTorque;

        if (i < 2)
            WCs[i].steerAngle = steer;
        else
            WCs[i].brakeTorque = brake;

        Quaternion quat;
        Vector3 position;
        WCs[i].GetWorldPose(out position, out quat);
        Wheels[i].transform.position = position;
        Wheels[i].transform.rotation = quat;
    }
}
```

KUVIO 14. Ote ajoskriptista

Ajoskriptiin tehtiin julkiset (public) parametrit kiihdytys- ja jarruteholle sekä eturenkaiden kääntöas-  
teelle. Julkiset parametrit näkyvät suoraan Unityssa, minkä ansiosta niiden arvojen vaihtaminen on  
nopeaa ilman, että skriptiä tarvitsee avata. Useiden kokeilujen jälkeen renkaiden ja ajettavuuden

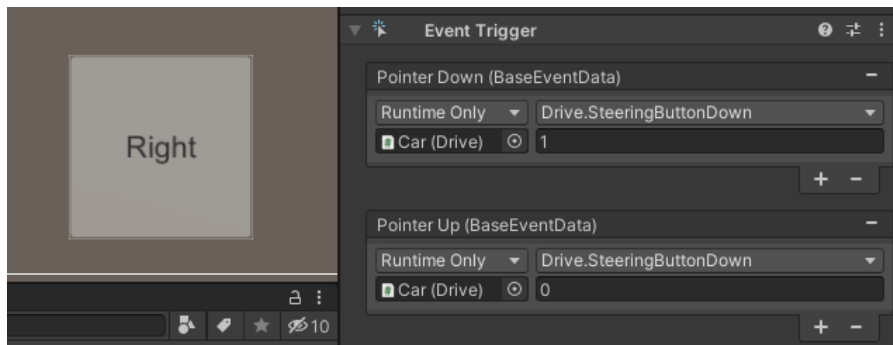
parametrit saatiin säädettyä hyvälle tasolle ja siirryttiin seuraavaan vaiheeseen – ajotien tekemiseen.

#### **4.6 Mobiiliohjaus**

Tämän prototyypin perimmäinen tarkoitus oli päästä testaamaan rallipelin pelattavuutta puhelimella. Tämän vuoksi peliin piti lisätä mahdollisuus ajaa autoa kosketusnäytöllä ja ratkaisutapoja toteutukseen oli monia. Unity tarjoaa uuden Input System -paketin monialustaohjausta varten, jolla peliin saadaan lisättyä tuki puhelimien lisäksi myös peliohjaimille. Jos peli on tarkoitus julkaista usealle eri alustalle, nopeuttaa Input System työtä huomattavasti, koska joka alustalle ei tarvitse luoda omaa ohjasskriptiratkaisua. (Unity Technologies 2021l.)

Toinen vaihtoehto, johon prototyypin osalta myös päädyttiin, oli hyödyntää näppäimistöohjaukseen luotua skriptiä ja määritellä ruudulle lisättävät näppäimet lähettämään samat komennot kuin näppäimistö. Käyttöliittymää varten Unity sisältää Canvas-elementin, johon voidaan lisätä nappeja, tekstiä sekä kuvia ja niitä voidaan liikuttaa ja järjestää vapaasti. Canvas-elementin napit ovat kattavia suoraan myös puhelimella, joten lisäämällä neljä nappia saatiin tarvittavat näppäimet puhelinohjausta varten.

Napeilla on Unityssa oletusmuoto valmiina ja napin kokoa ja väriä on mahdollisuus vaihtaa. Napin ulkomuotoa voi vaihtaa myös omalla Sprite-grafiikalla. Jotta napit saatiin oikeasti toimimaan, lisättiin niihin Event Trigger -komponentti, jonka avulla napille saadaan määriteltyä toiminnot klikkauksille. Oletustoimintona nappi on vain klikkauskohtainen, joten jokaisella klikkauksella voidaan tehdä toiminto, mutta koska pelissä tarvitaan pitkiä painalluksia, tehtiin Event Triggeriin omat toiminnot napin painamiselle ja napin nousulle. Jokaiselle napille määriteltiin polku oikeaan funktioon. Kaasu- tai jarrunappia painettaessa lähetetään ajoskriptiin parametri 1, ja kun nostetaan sormi napilta, lähetetään parametri 0. Sama toteutettiin myös kääntymiselle sillä erotuksella, että vasemmalle painettaessa lähetetään parametri -1. Nämä toiminnot lisäämällä kosketusohjaus toimi vastaavalla tavalla kuin näppäimistöohjaus (kuvio 15.).



KUVIO 15. Kosketusnäyttönäppäinten määrittely

## 4.7 Isometrinen kamera

Pelin kuvakulman on tarkoitus olla isometrinen, eli pelinäkymä on yläviistosta ja kamera seuraa auton liikettä. Unity luo kamerakomponentin aina projektiin, mutta oletuksena se on staattinen.

Kameran asemointia x-, y- ja z-akseleita muuttamalla saatiin haettua oikea kuvakulma ja etäisyys ohjattavaan autoon nähden (kuvio 18). Lisäksi luotiin lyhyt skripti, jonka avulla kamera seuraa auton liikettä säilyttäen saman kuvakulman ja etäisyyden (kuvio 16.).

```
public Transform car; //Muuttuja, johon tallennetaan viittaus auto-peliobjektista
private Vector3 offset; //Muuttuja, johon tallennetaan kameran ja pelaajan välinen etäisyys

@ Unity Message | 0 references
void Start()
{
    //Käynnistyksen yhteydessä lasketaan ja talletetaan muuttujaan auton ja
    //kameran välinen sijaintiero 3D-ympäristössä
    offset = transform.position - car.position;
}

//Päivitys
@ Unity Message | 0 references
void Update()
{
    // Päivitetään kameran sijaintia auton sijainnin mukaan ja
    // lisätään offset muuttujan arvot jotta kamera pysyy yläviistossa.
    transform.position = car.position + offset;
}
```

KUVIO 16. Kameraskripti

## 4.8 Ohjattavuuden hiominen

Kun pelin kaikki tarvittavat osa-alueet oli saatu toteutettua ja peli oli pelattavassa kunnossa, keski-tyttiin pelattavuuden ja auton ohjattavuuden hienosäätämiseen. Ensimmäinen huomiota vaativa

asia oli auton kääntyminen. Tehdyssä ajoskriptissä kääntyminen vasemmalle tai oikealle oli joko päällä tai pois, ja tämän tuloksena rengas kääntyi välittömästi määritellyyn kääntökulmaan nappia painettaessa. Tämä teki ohjauksesta herkän ja auton vaikeaksi hallita. Skriptiä muutettiin kääntyvyyden osalta siten, että nappia pohjassa pidettäessä eturenkaiden kääntökulma muuttui vähitellen kohti maksimikulmaa. Tämä toteutettiin lisäämällä kääntökulmaa päivitysfunkiossa if-lauseen avulla (kuvio 17.).

```
// Jos ohjausnappia on painettu on steer arvo joko -1 (vasen) tai 1 (oikea).  
// if-lauseella tarkastetaan kumpaan suuntaan renkaan pitää kääntyä ja  
// lisätään steeringAngle-muuttujaan (kääntökulma) arvo 1 jokaisessa päivityksessä.  
// Tämä kääntää renkaan kulmaa vaihteittain maksimikulmaa (maxSteeringAngle) kohti.  
if (steer < 0)  
{  
    if (steeringAngle < maxSteeringAngle)  
    {  
        steeringAngle += 1f;  
    }  
}  
else if (steer > 0)  
{  
    if (steeringAngle < maxSteeringAngle)  
    {  
        steeringAngle += 1f;  
    }  
}  
// Kun ei käännä, palautuu renkaat oletusasentoon eli suoraan eteenpäin.  
else if (steer == 0)  
    steeringAngle = 0;
```

KUVIO 17. Ote ajoskriptin update-funktiosta, jossa renkaan kääntökulma päivittyy

Toinen muutos tehtiin auton jousituksen pituuteen. Jousissa oli liikaa liikkumavaraa ja kun autolla ajoi hyppyyn, renkaat laskivat liian alas suhteessa auton koriin. Tämä oli enemmän visuaalinen ongelma, mutta liika pituus aiheutti myös jousien liiallisen löysyyden, mikä ilmeni tiukoissa käännöksissä auton liiallisena kallistumisena. Jousituksen säätäminen tapahtui renkaihin lisätyn Wheel Collider -komponentin arvoja muuttamalla. Arvoja on useita ja oikean suhteen löytymisen vaati paljon arvojen muuttamista ja testausta. Lopulta haluttu arvo löytyi, mutta kääntökallistuksen korjaaminen vaati myös renkaiden sivuttaiskitka-arvojen muokkausta, joka tapahtui myös Wheel Collider -komponentin kautta.

Lopuksi auton kiihdytykselle ja jarrutusteholle haettiin hyvät arvot. Koodausvaiheessa määritelty kiihtyvyyssarvo oli liian pieni ja tästä johtuen auto kiihtyi liian hitaasti. Koska kiihtyvyyttä lisättiin, myös jarrutustehoa oli tarvetta hieman nostaa. Näiden arvojen säätäminen ja oikean tasapainon hakeminen onnistui suoraan Unityn työnäkymästä, koska säädettävät ajoskriptin muuttujat oli määritelty julkisiksi.



## 4.9 Testaus ja lopputulos

Ohjelmointivaiheessa prototyyppiä testattiin tietokoneen lisäksi myös älypuhelimella, joka oli mahdollista Unity Remote -sovelluksen avulla. Sovellus ladattiin testipuhelimeen Play-kaupasta ja puhelimen kytkettiin USB-kaapelilla kiinni tietokoneeseen. Kun sovellus oli puhelimesta käynnissä ja peli ajettiin Unitylla, pelinäköymä oli testattavissa suoraan puhelimen näytöltä. Teknisesti peli pyöritetään tietokoneelta ja peilataan USB-kaapelia hyödyntäen puhelimeen, joten viivettä kosketusnäyttöpainallusten välittymisessä oli havaittavissa, eikä tarkempi ajettavuuden testaaminen ollut mahdollista Unity Remoten avulla. Yleistä pelinäköymää ja näppäinten kosketusnäppäinten toimintojen testaamista tämä kuitenkin helpotti ja nopeutti huomattavasti.

Lopullista prototyyppitestausta varten tehtiin Unitylla Android APK eli suoraan puhelimeen asennettava ohjelma. Valmis APK siirrettiin puhelimen muistiin ja asennuksen jälkeen se voitiin käynnistää, kuten muutkin Android-sovellukset. Tämä mahdollisti prototyypin paremman testaamisen ilman viivettä (kuvio 18.).



KUVIO 18. Prototyyppi käynnissä Android-puhelimessa

Prototyypin avulla oli tarkoitus testata peli-idean toimivuutta mobiililaitteella ja tämä tavoite saavutettiin. Peliä päästiin testaamaan useassa eri laitteessa ja näyttökoossa. Tärkein osa-alue, eli ohjattavuus toimi oikein hyvin kosketusnäytön virtuaalinäppäimillä ja ajaminen oli miellyttävää. Lisäksi isometrinen kuvakulma toimi hyvin myös pienemmällä näytöllä pelatessa. Valmiit resurssipaketit,



joita hyödynnettiin graafisen ilmeen luontiin, antoivat myös hyvän yleiskuvan peliin halutusta visuaalisesta tyylistä.

Prototyyppi rohkaisi jatkamaan peli-idean toteutusta. Pelin jatkokehityksen seuraavassa vaiheessa keskitytään enemmän pelillisiin ominaisuuksiin, kuten ajan ottoon, tekoälyvastustajien aikojen toteutukseen, vauriomallinnukseen ja myös omien 3D-mallien tekemiseen.

## 5 POHDINTA

Ennen opinnäytetyön aloitusta minulla oli jonkin verran kokemusta pelin tekemisestä, Unitysta ja koodaamisesta oman harrastuneisuuden ja pelikurssin kautta. Prototypoinnista ja pelisuunnittelun vaiheista kokemukseni oli kuitenkin hyvin vähäistä ja tiesin, että haasteena tulee olemaan peli-idean tärkeimpien ominaisuuksien valinta toteutettavaan prototyyppiin ja rajausten säilyttäminen läpi toteutuksen. Aiemmasta kokemuksestani olin huomannut, että päätetyissä raameissa pysyminen on yllättävän hankalaa, ja projekti alkaa paisua helposti lisätoteutuksilla, jotka taas hidastavat kehitystyötä.

Ajatuskartta (liite 1.), jossa peli-ideaa kuvaillaan, on laaja ja sisältää suuren osan lopulliseen peliin halutuista ominaisuuksista. Rajausta onnistui kuitenkin hyvin, koska ensimmäisen prototyypin tarkoitus oli testata, toimisiko peli mobiililaitteella ohjattavuuden ja pelattavuuden suhteen. Yleiseen prototyyppiin tutustumisesta työn alussa oli suurta hyötyä oman prototyypin toteutuksessa ja varsinkin ominaisuuksien rajaamisessa.

3D-mallintamisesta minulla oli hyvin vähän kokemusta, ja tästä johtuen valmiiden resurssipakettien hyödyntäminen oli välttämätöntä. Unityn suosion hyvä puoli tulikin tässä hyvin esille, koska ilmaisia resursseja tekstuurien ja 3D-mallien osalta löytyy valtavasti, eikä kaikkea tarvinnut osata tehdä itse. Tämän ansiosta prototyypin rakentaminen oli tehokasta ja keskittyminen haluttuihin ominaisuuksiin oli mahdollista.

Koko opinnäytetyö ja peliprototyypin kehittäminen antoivat paljon tietotaitoa prototypoinnista ja peliprototyypin suunnittelusta sekä toteutuksesta. Prototyyppi onnistui hyvin ja sen avulla todettiin, että peliä kannattaa lähteä jatkokehittämään suunnitelman mukaisesti.

## LÄHTEET

4art Media 2021. Character Concept Art Services by 3D Game Art Studio, Denton – Texas. Hakupäivä 12.8.2021. [Character Concept Art Services by 3D Game Art Studio, Denton - Texas - 4art.com](https://4art.com).

Babich, Nick 2020. Everything You Need to Know About Wireframe Design and Prototypes. Hakupäivä 16.7.2021. [Wireframe Design & Prototype Must-Knows | Adobe XD Ideas](https://www.adobe.com/xd/ideas/wireframe-design-prototype-must-knows).

Bennett, Darren 2020. Prototyping: the mother of invention. Hakupäivä 02.09.2021. [Prototyping: the mother of invention | by Darren Bennett | UX Collective \(uxdesign.cc\)](https://uxdesign.cc/prototyping-the-mother-of-invention-by-darren-bennett).

Creately 2021. The Easy Guide to Prototyping | Prototyping Types and Process. Hakupäivä 13.7.2021. [The Easy Guide to Prototyping | Prototyping Types and Process \(creately.com\)](https://creately.com/the-easy-guide-to-prototyping).

Cui, Sunny 2018. A Guide to Paper-Prototype Sketching. Hakupäivä 13.7.2021. [A Guide to Paper-Prototype Sketching | by Sunny Cui | Medium](https://medium.com/@sunnycui/a-guide-to-paper-prototype-sketching).

DeerStudios 2017. Update v0.1 11/6/17 (Prototype Build). Hakupäivä 11.8.2021. [Update v0.1 11/6/17 \(Prototype Build\) - Squadron - Multiplayer FPS \(Unity 3D\) by DeerStudios \(itch.io\)](https://itch.io/game/116617/prototype-build).

Fitzgerald, Ryan 2021. What is Concept Art. Hakupäivä 11.10.2021 [What is Concept Art? | Job Role & Salary Expectations | CG Spectrum](https://www.cg-spectrum.com/what-is-concept-art/).

Fullerton, Tracy 2014. Game Design Workshop: A Playcentric to Creating Innovative Games. New York: CRC Press.

Grassi, Claudio 2021. Unity Game Development — From Prototype to Work of Art. Hakupäivä 04.8.2021. [Unity Game Development — From Prototype to Work of Art | by Claudio Grassi | Medium](https://medium.com/@claudiograssi/unity-game-development-from-prototype-to-work-of-art).

Incredibuild 2021. What is Visual Studio. Hakupäivä 10.11.2021. [What is Visual Studio? - Incredibuild](https://www.incredibuild.com/what-is-visual-studio).

Indeed 2021. 10 Types of Prototypes. Hakupäivä 24.9.2021. [10 Types of Prototypes \(With Explanations\) | Indeed.com](#)

Martin, Matthew 2021. Prototyping Model in Software Engineering: Methodology, Process, Approach. Hakupäivä 24.10.2021. [Prototyping Model in Software Engineering: Methodology, Process, Approach \(guru99.com\)](#).

NASA 2021. Overview of the Wright Brothers Invention Process. Hakupäivä 07.10.2021. [Overview of the Wright Brothers' Invention Process \(nasa.gov\)](#).

Petty, Josh 2021. What is Unity 3D & What is it Used For? Hakupäivä 7.11.2021. [What is Unity 3D & What is it Used For? \(conceptartempire.com\)](#).

Sampathi, Nitin 2018. Prototyping for Virtual Reality. Hakupäivä 29.8.2021. [https://blog.prototypr.io/prototyping-for-virtual-reality-a1036767dd45](#).

Treehouse 2014. How I Rapidly Prototype Websites. Hakupäivä 12.8.2021. [How I Rapidly Prototype Websites \[Article\] | Treehouse Blog \(teamtreehouse.com\)](#).

Unity Technologies 2021a. Unity Gaming Services. Hakupäivä 7.11.2021. [Grow a successful business with Unity Gaming Services | Unity](#).

Unity Technologies 2021b. Quick guide to the Unity Asset Store. Hakupäivä 7.11.2021. [Quick guide to the Unity Asset Store - Unity \(unity3d.com\)](#).

Unity Technologies 2021c. Visual Studio C# integration. Hakupäivä 10.11.2021 [Unity - Manual: Visual Studio C# integration \(unity3d.com\)](#).

Unity Technologies 2021d. Unity Remote. Hakupäivä 07.11.2021. [Unity - Manual: Unity Remote \(unity3d.com\)](#).

Unity Technologies 2021e. Car'Toon: The Sport Car with interior. Hakupäivä 12.8.2021. [Car'Toon : The Sport Car with interior | 3D Land | Unity Asset Store](#).

Unity Technologies 2021f. Unity Documentations, Rigidbody. Hakupäivä 24.11.2021. [Unity - Scripting API: Rigidbody \(unity3d.com\)](#).

Unity Technologies 2021g. Unity Documentations, Box Collider. Hakupäivä 24.11.2021. [Unity - Manual: Box Collider \(unity3d.com\)](#).

Unity Technologies 2021h. Unity Documentations, Wheel Collider. Hakupäivä 24.11.2021. [Unity - Manual: Wheel Collider \(unity3d.com\)](#).

Unity Technologies 2021i. Unity Documentations, Terrain. Hakupäivä 27.11.2021. [Unity - Manual: Terrain \(unity3d.com\)](#).

Unity Technologies 2021j. FREE Stylized PBR Textures Pack. Hakupäivä 27.11.2021 [FREE Stylized PBR Textures Pack | 2D Textures & Materials | Unity Asset Store](#).

Unity Technologies 2021k. Low-Poly Simple Nature Pack. Hakupäivä 27.11.2021. [Low-Poly Simple Nature Pack | 3D Landscapes | Unity Asset Store](#)

Unity Technologies 2021l. Input System. Hakupäivä 11.12.2021. [Input System | Input System | 1.0.2 \(unity3d.com\)](#).

Wright-brothers.org 2011. Wright Kites and Gliders. Hakupäivä 07.10.2021. [Wright Kites and Gliders \(wright-brothers.org\)](#).

Velling, Andreas 2020. What Is Rapid Prototyping? Hakupäivä 18.9.2021. <https://fractory.com/what-is-rapid-prototyping/>.

Victorino, Laurent 2015. What you should know about game prototypes. Hakupäivä 22.7.2021. [What you should know about game prototypes – I make games for food \(lvictorino.com\)](#).

Zmorph SA 2016. Designing a 3D Printed Wireless Mouse. Hakupäivä 10.8.2021. [Designing a 3D Printed Wireless Mouse | by Zmorph SA | Medium](#).

