IZRADA 3D MODELA SJEDINJENIH NA WEB STRANICI U SVRHU KORIŠTENJA ZA DIZAJNIRANJE SKATE PARKOVA

Bezi, Sven

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

IZRADA 3D MODELA SJEDINJENIH NA *WEB* STRANICI U SVRHU KORIŠTENJA ZA DIZAJNIRANJE *SKATE* PARKOVA

Sven Bezi

Zagreb, lipanj 2018.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 20.6.2018.

Sven Bezi

Predgovor

Zahvaljujem mentoru red. prof. dr. sc. Predragu Šuki koji je pratio cijeli proces nastajanja diplomskog rada i svojim savjetima i entuzijazmom usmjerivao me kako da prevladam probleme koji bi se pojavili pri izradi diplomskog rada.

Sažetak

Ovaj će se završni rad baviti procesom izrade *web*-stranice koja služi kao reprezentativna platforma i procesom izrade desktop aplikacije koja unutar sebe sadržava teksturirane 3D objekte u svrhu dizajniranja *skate*-parkova.

Prema mojem istraživanju ovakvu vrstu 3D aplikacije još nisam susreo te bi ona uvelike mogla unaprijediti *skateboarding* i potaknuti širenje skejtbordinške kulture.

Cilj je ovoga projekta napraviti alat koji se lako uporabljuje i koji dostavlja kvalitetne rezultate prijenosom ideje na konkretnu vizualizaciju, koja se poslije može iskoristiti kao temelj pregovora oko izgradnje *skate*-parkova.

Ključne riječi: 3D modeliranje, teksturiranje, *web*-dizajn, *web development*, *Unity*, *game development*

Abstract

This final work will be done through the process of creating a web site that serves as a reputable platform and a desktop application development process that contains textured 3D objects inside it for the purpose of designing skate parks.

In my research of my this kind of 3D application I have not met yet and could greatly improve skateboarding and encourage the spread of skateboarding culture.

The aim of this project is to create a tool that is easy to use and delivering quality results by passing ideas to concrete visualization, which can later be used as the basis for negotiations on the construction of skate parks.

Key words: 3D modeling, texturing, web design, web development, Unity, game development.

Sadržaj

1. Uvod

Tema je ovog rada proces izrade alata (aplikacije) namijenjena korisnicima zainteresiranih za izradu dizajna *skateparka* koji dizajneri u procesu svog poslovanja mogu prilagati klijentima s kojima posluju, te na temelju te vizualizirane ideje može započeti gradnja *skateparka*.

Proces izrade ovog alata sadržava modeliranje *skate*-elemenata, teksturiranje, prezentaciju teksturiranih 3D modela, izradu strukture samog alata, izradu instalacijske datoteke alata te izradu reprezentativne *web*-stranice koja predočuje projekt te sadržava instalacijsku datoteku i osnovne informacije o alatu i kako se uporabljuje.

Web-stranica također sadržava opciju korisnika alata da sami predlože nove ideje *skate*-elemenata, šaljući svoje skice preko kontakt forme te samim time šire asortiman *skate*-elemenata ponuđenih u programu.

2. Modeliranje *skate*-elemenata

Alat koji sam napravio zove se *Skate park builder*, a njegova najvažnija funkcija jest pozicioniranje 3D modela na plohi kako bi se došlo do željenog rezultata, savršenog dizajna.

Stoga sam modeliranje svih tih elemenata odabrao kao prvi korak.

Prije samog modeliranja potrebno je:

- definirati *skate*-elemente
- prikupiti referencije
- izrada vlastitih referencija.

Spreman za početak modeliranja, odlučio sam da će u prvoj verziji alata biti ponuđen dvadeset jedan element koji su, prema mojem mišljenju u današnje vrijeme najpopularniji i najfunkcionalniji za *skate*-parkove.

2.1. Definiranje *skate*-elemenata

Kada bi se ostali sportovi uspoređivali sa *skateboardingom*, možemo reći da je skejtpark mjesto za vježbanje i natjecanje skejtera kao što je nogometno igralište mjesto za vježbanje i natjecanje nogometaša. No, za razliku od ostalih sportskih terena, skejtparkovi su uvijek različiti, sadržavaju različite elemente i drukčije pozicije samih elemenata. Svaki skejtpark daje unikatno iskustvo svojim korisnicima. Skejtparkove možemo podijeliti u 3 vrste.

- *Bowl park* – daje korisnicima iskustvo *pool skateboardinga*[1]. To su parkovi u kojima korisnik uopće ne treba skidati nogu sa *skatea* kako bi vozio tim parkom
- *Street plaza park* – To su parkovi čiji su elementi replike uličnih elementa na kojima se također prakticira *skateboarding* (stube, rukohvati, klupe…).
- *Flow park* – kombinacija je *bowl parka* i *street plaza parka.* Ova se vrsta parkova najčešće upotrebljava na najvećim skejtboardinškim natjecanjima.

[1] *Pool skatebaording* – vožnja ispražnjenim betonskim bazenima 1970. godine. Još se naziva i „tranzicijskim *skateboardingom*".

Slika 2-1. *Bowl skatepark*

Slika 2-2. *Street plaza skatepark*

Slika 2-3. *Flow skatepark*

Budući da su *Flow skateparkovi* najzastupljeniji, odlučio sam se za modeliranje elemenata koji se najviše tiču te vrste parka.

2.1.1. Definiranje vrste elementa

Skate-elemente možemo podijeliti u više kategorija. Elementi se razlikuju po tome koji se trikovi na njima mogu izvoditi. Kategorija trikova koje se mogu izvoditi na *skateu* jesu sljedeće:

- *Grabs*
- *Flip tricks*
- *Slides and grinds*
- *Plants*
- *Manuals.*

Stoga treba pažljivo dizajnirati *skatepark* kako bi svi trikovi mogli biti prakticirani na određenim elementima. Elemente dijelimo ovako:

- Pipes – Qouter pipe, Half pipe, Spine, Bowl
- Kickers – Regular kicker, Curved kicker/launcher, Hip, Pyramid
- Rails – Hand rail, Square rail, Round rail, Rainbow rail,
- Manual pads
- Grind boxes
- Funboxes.

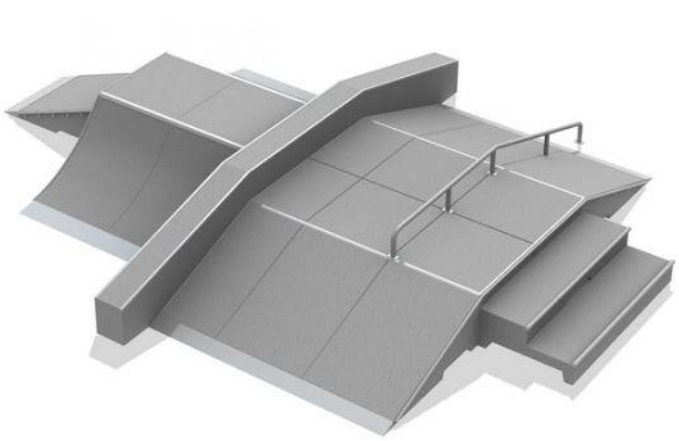
Slika 2-4. Skejter prakticira trik na HALF PIPE-u

Slika 2-5. *Curved kicker/launcher*

Slika 2-6. Skejter prakticira trik na HAND RAIL-u

Slika 2-7. Skejter prakticira trik na MANUAL PAD-u

Slika 2-8. Skejter se sprema za trik na GRIND BOX-u



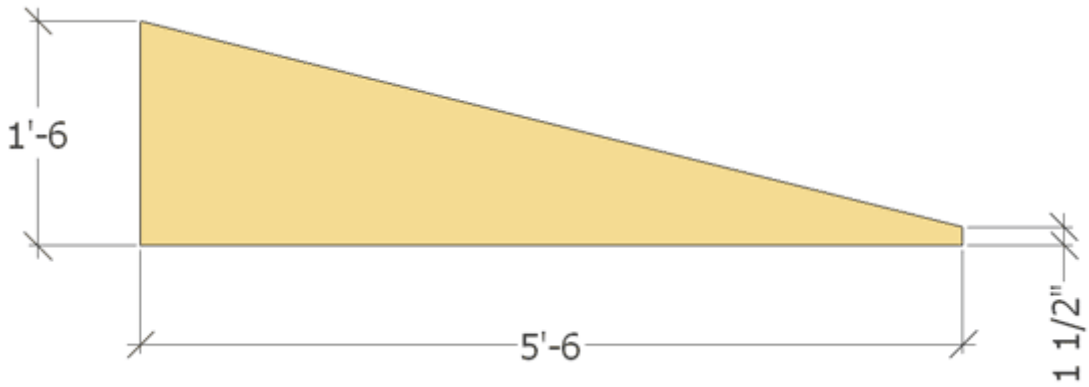
Slika 2-9. FUNBOX *skate*-element

Navedeni *skate*-elementi esencijalni su dijelovi *skateparka* koji omogćuju skejterima da izvode sve od navedenih kategorija trikova. Elementi mogu biti napravljeni u raznim dimenzijama i dolaziti u raznim oblicima, i sve je dopušteno dokle god ispunjavaju svoju funkcionalnost.

2.1.2. Definiranje dimenzija *skate*-elemenata

Prije samog modeliranja potrebno je uzeti stvarne mjere *skate*-elemenata kako bi se te brojke mogle prenijeti u program za modeliranje i napraviti element u stvarnim dimenzijama. Ako elementi nisu modeliranu i istom omjeru, pojavit će se problem pri unosu/importu u druge programe gdje se tim modelima želim koristiti.

Na temelju istraživanja došao sam do zaključka da se trebam koristiti dimenzijama pojedinih elemenata s *web*-stranice DIY Skate[2], gdje sam pronašao mjere u inčima te sam ih zatim pretvorio u centimetre kako bih ih mogao unijeti u program za 3D modeliranje u kojemu sam radio modeliranje.



Slika 2-10. Bokocrt elementa s dimenzijama kojima sam se koristio za modeliranje

2.1.3. Definiranje okvirne cijene na temelju istraživanja

Cijene do kojih sam došao na temelju istraživanja ne uključuju ništa više osim samog materijala za izradu određenog elementa. Budući da sam išao na varijante betonskih *skate*-parkova od materijala se uporabljuju: beton, željezne šipke i armaturna mreža za pojedine

[2] http://www.diyskate.com – stranica koja sadržava savjete za izradu *skate*-elemenata

elemente. Na temelju cijena tih materijala odredio sam konačnu cijenu izrade. Jednostavna formula za izračun cijene elementa:

volumen betona koji je potreban za formiranje elementa + šipka ili metalni rub + armaturna mreža.

Primjer računanja cijene *kickera* (slika 2-10.):

ŠIRINA – 182,88 cm

DUŽINA – 100 cm

VISINA – 60,96 cm

širina * visina * dužina = volumen

182,88 * 100 * 60,96 = 1 114 836,48 cm^3 / 2 = 557 418,24 cm^3

Za ovaj element nije potrebno koristiti se niti armaturnom mrežom ni šipkom/metalnim rubom pa stoga računamo samo cijenu betona koja je potrebna za izgradnju.

Kubni centimetri pretvoreni u kilograme:

557 418,24 cm^3 / 1000 = 557,41 kg.

Cijena betona – 35 kg = 4 USD.

Cijena 1 kg betona = 0,11 USD.

KONAČNA CIJENA MATERIJALA ELEMENTA: 557,41 * 0,11 = 61,27.

CIJENA MATERIJALA ZA IZRADU *KICKERA* IZNOSI OKO 60 DOLARA NA INOZEMNOM TRŽIŠTU.

NA HRVATSKOME TRŽIŠTU CIJENA IZRADE *SKATE*-ELEMENTA TIH DIMENZIJA BILA BI OKO 800 KUNA. CIJENA JE DOBIVENA IZRAČUNOM CIJENA MATERIJALA U PRODAJNOM CENTRU „Ikoma“.

2.2. Prikupljanje referencija za modeliranje *skate*-elemenata

Ovaj korak u procesu izrade alata svodi na prikupljanje oblika elemenata koji će služiti kao referencija po kojoj modeliram 3D objekt *skate*-elementa te izrađujem oblik u vektorskoj grafici. Istraživanja i prikupljanje besplatnih slika bili su mi dobar temelj za izradu vektora.

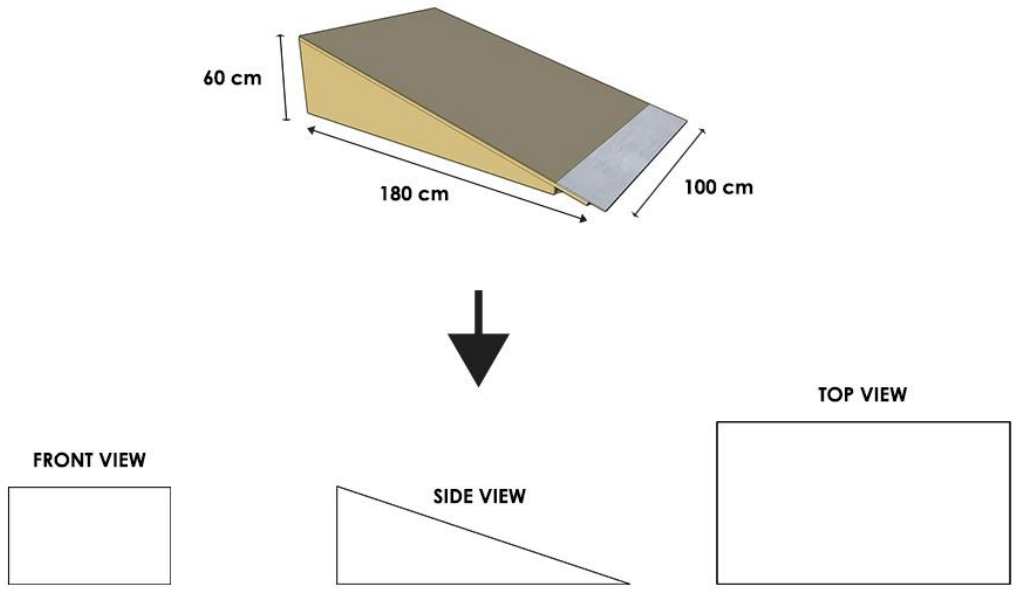
Vrste elemenata kojima sam se koristio za izradu skica jesu: Pipes, Kickers, Rails, Manual pads i Funboxes.

2.2.1. Prikupljanje informacija o dimenzijama *skate*-elemenata

Kako bi svi 3D modeli unutar alata izgledali dosljedno jedan drugom, potrebno je prije modeliranja definirati realne mjere tih elemenata, a mjere određenih elemenata uzimao sam s već spomenute stranice DIYSkate. Ostale elemente koje sam modelirao koristeći se svojim iskustvom i kreativnošću prilagođivao sam realnim mjerama već modeliranih elemenata čije sam dimenzije prikupio. Rezultat modeliranja elemenata po proporcionalnim mjerama omogućio mi je lakše uvođenje objekata u program za izradu aplikacije te naknadno nisam morao prilagođivati dimenzije. Sve su mjere bile precizne.

2.3. Izrada vlastitog materijala

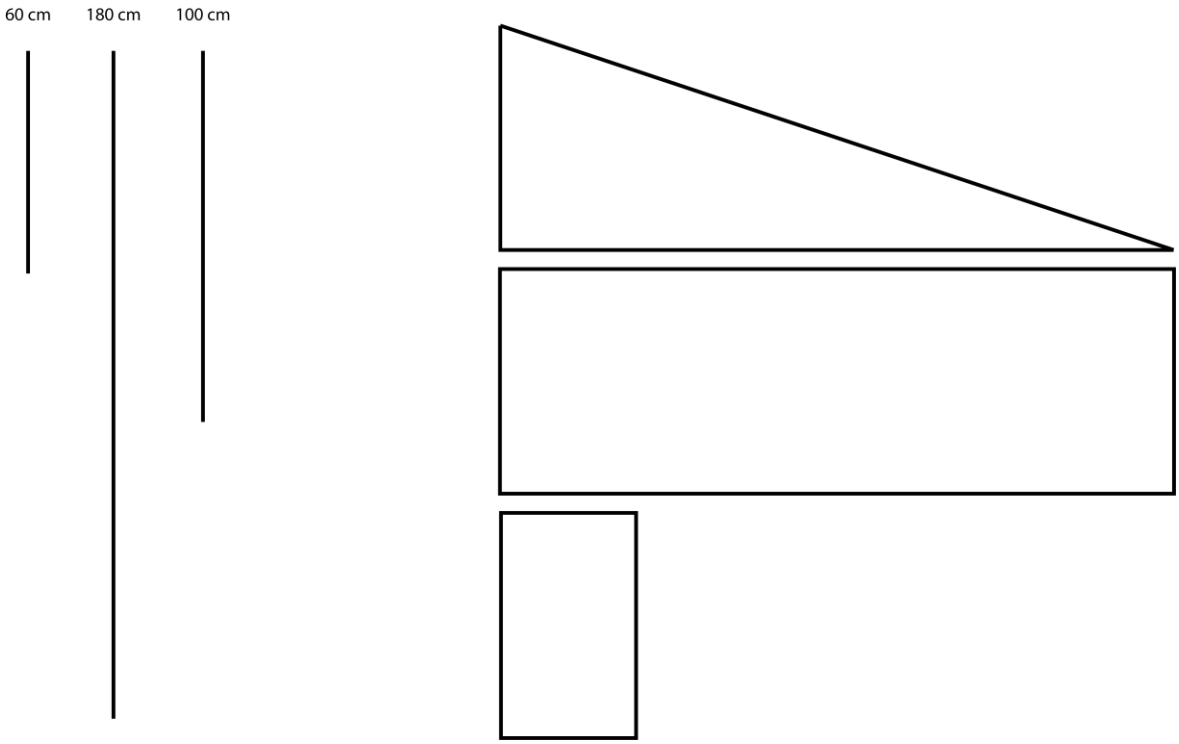
Na temelju fotografija i ostalih referencija koje dolaze u trodimenzionalom obliku mogu započeti rad na modeliranju, ali prvo iz tih fotografija moram izvući ortografsku skicu *front*, *top* i *side* pogled (odnosno nacrt, tlocrt i bokocrt) kako bih si olakšao modeliranje i što preciznije izradio *skate*-element.



Slika 2-11. Ortografske skice izrađene prema referenciji

2.3.1. Izrada ilustracija vektorskom grafikom kao referencija za *skate*-model

Ilustracije sam izrađivao u programu Adobe Illustrator CC 2017, a rađene su u svrhu ortografskog pogleda iz različitih perspektiva na element. Kako bi 3D model bio što preciznije modeliran, uporabljene su mjere sakupljene u procesu istraživanja. Mjerama tih elemenata također se koristim u programu za modeliranje kao stvarnim dimenzijama objekta.



Slika 2-12. Mjere i skice ortografskih pogleda na element

2.4. Izrada 3D *skate*-elemenata

U ovom dijelu procesa napokon dobivamo trodimenzionalne rezultate vektorskih ortografskih 2D skica. Prije samog početka potrebno je konfigurirati program kako bismo sami sebi olakšali posao modeliranja, a zatim unosimo referentne ilustracije koje smo napravili u prethodnom koraku. Potom slijedi sam proces modeliranja i na kraju priprema modela za teksturiranje i izvoz modela u FBX formatu kako bi bio teksturiran u programu za teksturiranje.

2.4.1. Definiranje postavki programa za rad

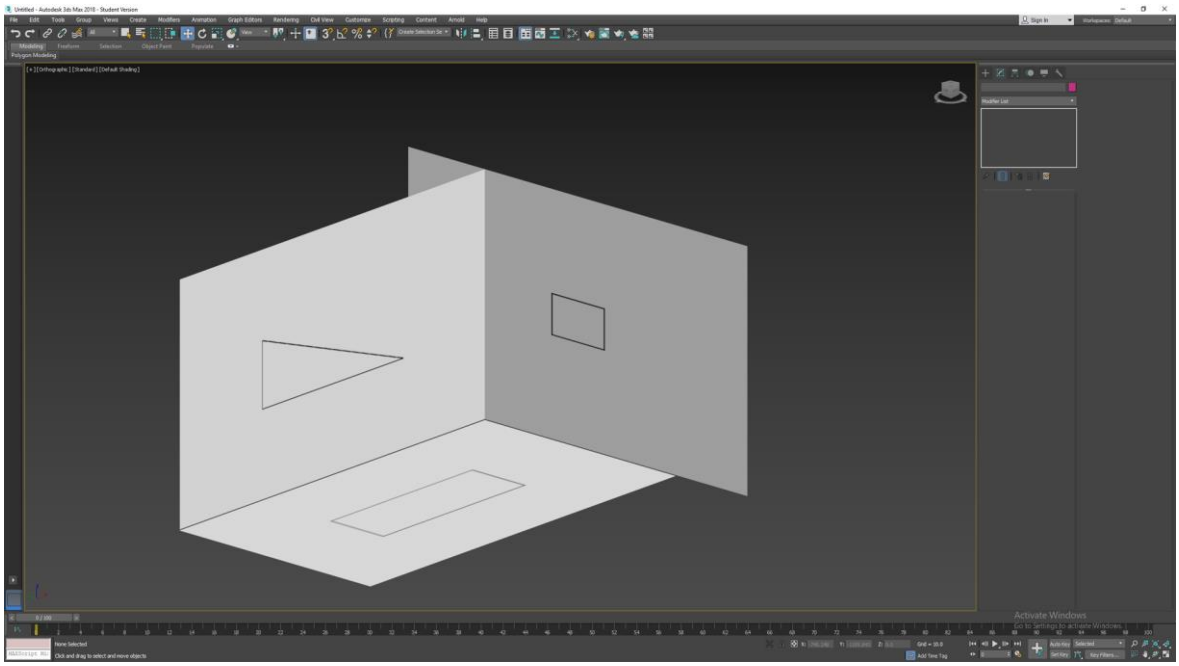
3D modeliranje rađeno je u programu 3DS max 2018. Pokretanjem programa dobivamo predefinirane postavke programa koje moramo izmijeniti kako bismo lakše i preciznije izradili 3D model. Prva stvar koju moramo promijeniti jest perspektivni pogled na scenu, a taj pogled prebacujemo na ortografski koji nam daje precizniji pogled na model bez perspektivnog skošenja koji uvelike otežava proces modeliranja. Zatim kreiramo 3 *plane*

objekta koji mi služe za aplikaciju ilustracija koje sam izradio u prethodnom koraku. *Plane* objekti s ilustracijama razmješteni su ovako: jedan ispod samog objekta, drugi s prednje strane i treći s bočne strane objekta. *Plane*-objekte trebali bismo „zaleditiˮ kako slučajno ne bismo utjecali na njih pri izradi modela. Kada bismo ih slučajno pomaknuli, naš 3D model mogao bi ispasti pogrešnih proporcija.

Plane-objekti moraju biti istih proporcija kako ne bi došlo do rastezanja ili suzivanja slike, što također utječe na naš konačni 3D model.

2.4.2. Unos referentnih ilustracija za modeliranje

Referentne ilustracije koje smo u prethodnom koraku spremili u JPG formatu, jednostavno odvlačimo na odgovarajuće *plane*-objekte kako bismo stvorili njihov prikaz na tim objektima i u konačnici stvorili podloge za rad i modeliranje. Naša je scena sada spremna za modeliranje.



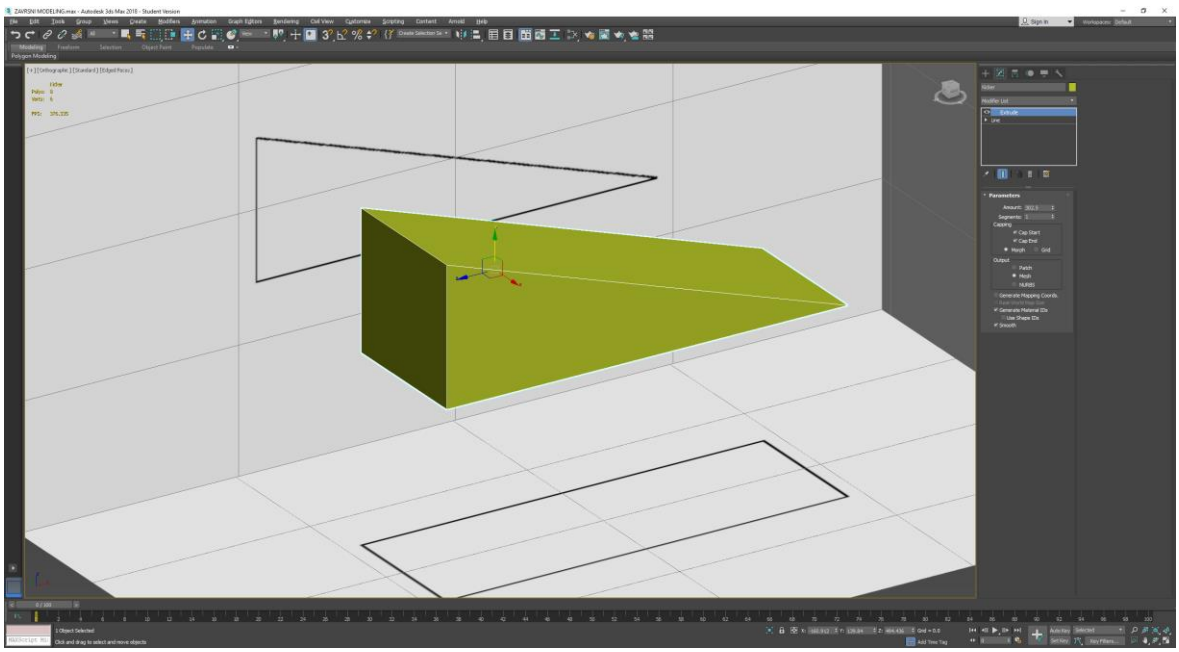
Slika 2-13. Scena načinjena od 3 *plane*-objekta s apliciranim ilustracijama

2.4.3. Modeliranje *skate*-elemenata

Modeliranje počinje odabirom jednog od ortografskih pogleda na scenu (nacrt, tlocrt i bokocrt). Ja sam izabrao bokocrt, a zatim sam uzeo *Line tool* kako bih nacrtao oblik *kickera*. Kada sam dobio 2D skicu s pomoću Line toola, dodao sam *line* objektu *modifier*

Extrude. *Extrude modifierom* stvaramo 3D objekt iz skice koje smo napravili *Line toolom*. Potom se prebacujem u tlocrt kako bih precizno napravio *Extrude* prema referenciji koja se nalazi ispod. Oblik *kickera* je gotov.

3D model *skate*-elementa *kicker* sadržava osam poligona, što je najmanji broj kako bi ovaj oblik bio kreiran u 3D svijetu. Moramo dobro paziti na to da imamo što manji broj poligona kako ne bi utjecalo na naš alat u koji ćemo unijeti, ali da opet nema premalo poligona kako objekt ne bi gubio vizualnu kvalitetu.

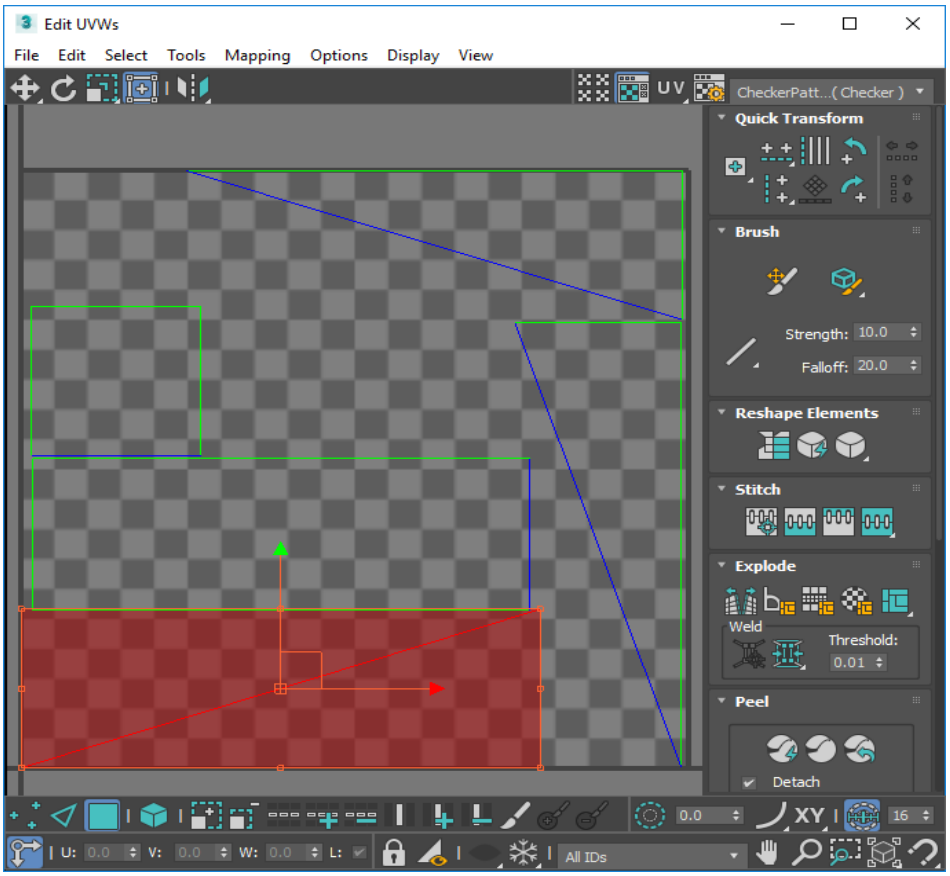


Slika 2-14. 3D objekt izrađen prema vektorskim referencijama

2.4.4. Priprema 3D modela za teksturiranje

Kada smo zadovoljni izrađenim 3D objektom, ostaje nam još jedan korak prije spremanja i teksturiranja samog objekta, a to je dodavanje *modifiera* „Unwrap UVW“. Taj *modifier* omogućuje nam da stvorimo koordinate na 3D objektu po kojima ćemo moći stavljati i pozicionirati teksture.

Taj *modifier* zapravo nam omogućuje da 3D objekt „razmotamo“ na ravnu plohu i posložimo poligone kako smatramo da nam je lakše raditi. Kada je Unwrap UVW apliciran na model, spremni smo spremiti scenu i napraviti *export* odgovrajućeg objekta. Objekt koji izvozim za teksturiranje ima ekstenziju FBX.



Slika 2-15. Unwrap UVW 3D modela *kickera*

3. Teksturiranje 3D *skate*-elemenata

Kada smo završili proces modeliranja za ostalih 20 *skate*-elemenata, spremni smo te 3D objekte unijeti u program za teksturiranje. Program kojim sam se koristio za teksturiranje zove se *Substance Painter*. Prije samog teksturiranje 3D modela potrebno je prvo konfigurirati program i prilagoditi ga sebi i svojim potrebama kako bismo što lakše radili. Kada unesemo objekt i krenemo teksturirati, moramo napraviti podjelu na objektu ako imamo više materijala. Za moj projekt potreban mi betona, a za neke elemente i metal. Zatim kreće sami proces apliciranja tekstura na određene dijelove modela. Ako želimo detaljniji izgled, možemo na model dodati i fine detalje. Kada smo zadovoljni teksturama, radimo *export* za daljnji proces.

3.1. Definiranje postavki programa za rad

Substance painter ima predefinirane postavke pregleda modela koje se sastoje od 2 prozora, a to je *3D/2D view*, gdje se na *3D viewu* prikazuje objekt koji smo importali u programa, a na *2D viewu* prikazuje se „Unwrap“ tekstura po kojima možemo crtati te ih modificirati kako bismo došli do željenog rezultata. Mojem radu odgovara samo *3D view* pa stoga isključujem *2D view* kako bih imao što veći i bolji pogled na sam 3D objekt pri aplikaciji tekstura i crtanju finih detalja na teksture, odnosno 3D objekt. Važni su nam pomoćni prozori koji nam daju opcije za rad na objektu. Četiri važna prozora kojima koristimo u procesu jesu:

- Layers
- Texture settings
- Texture Set list
- Properties.

Layers je prozor koji nam daje opcije da napravimo više slojeva tekstura koje apliciramo na objekt. Alatom *Layers* možemo dodavati više slojeva na kojima radimo (*layer*), možemo grupirati te slojeve u foldere kako bi ostali što bolje organizirani, možemo dodavati *Fill layere* koji nam omogućuju da stavimo određenu teksturu na taj posebni *layer*, možemo rabiti *Smart material*, što je zapravo kombinacija *Fill layera*, *Paint layera* i

ostalih efekata. *Paint* ili obični *layer* omogućuje nam crtanje raznim oblicima kistova po teksturama kako bismo dodali još više efekata, kao što su *roughness* i *height*. *Roughness* nam omogućuje da bojimo dio objekta na koji neće utjecati refleksija, a *height* nam daje vizualno izbočenja i udubljenja objekta kako bismo ostvarili 3D efeket na modelu bez dodatnog modeliranja, što je jako veliki plus jer tako možemo zamaskirati dijelove modela koje bi inače stvorili više poligona da ih tako modeliramo, a to bi utjecalo na performanse igre/aplikacije koju izrađujemo.

Texture Set list jest alat koji nam pomaže da pri importu objekta razložimo objekt na manje cjelina ako smo ga prije toga u *modeling* programu podijelili na određene cjeline. Objekt može biti podijeljen na manje skupine poligona tako da skupini poligona dodijelimo određeni ID indetifikator koji možemo nazvati kako god želimo. Na mojem primjeru modeliranja *Funboxa* moj se objekt sastoji od betonskog dijela *skate*-elementa i metalnog dijela elementa i stoga sam ja taj objekt modelirao kao jedan, ali sam ga naknadno podijelio i dodijelio mu 2 ID identifikatora. Prva skupina poligona rezervirana je za betonski dio *skate*-elementa, a druga skupina rezervirana je za metalni dio *skate*-elementa. Kada takav objekt unesem u *Substance painter*, imam mogućnost odabira *Texture Seta* na kojem želim trenutačno raditi.

Texture settigns nudi nam opciju „Baking“. Ta opcija omogućuje nam da izvučemo mape za teksturiranje na objektu po primjeru uvezenog 3D objekta u program. Inače, ako ne odradimo „Baking“, ne možemo se koristiti *Smart* maskama.

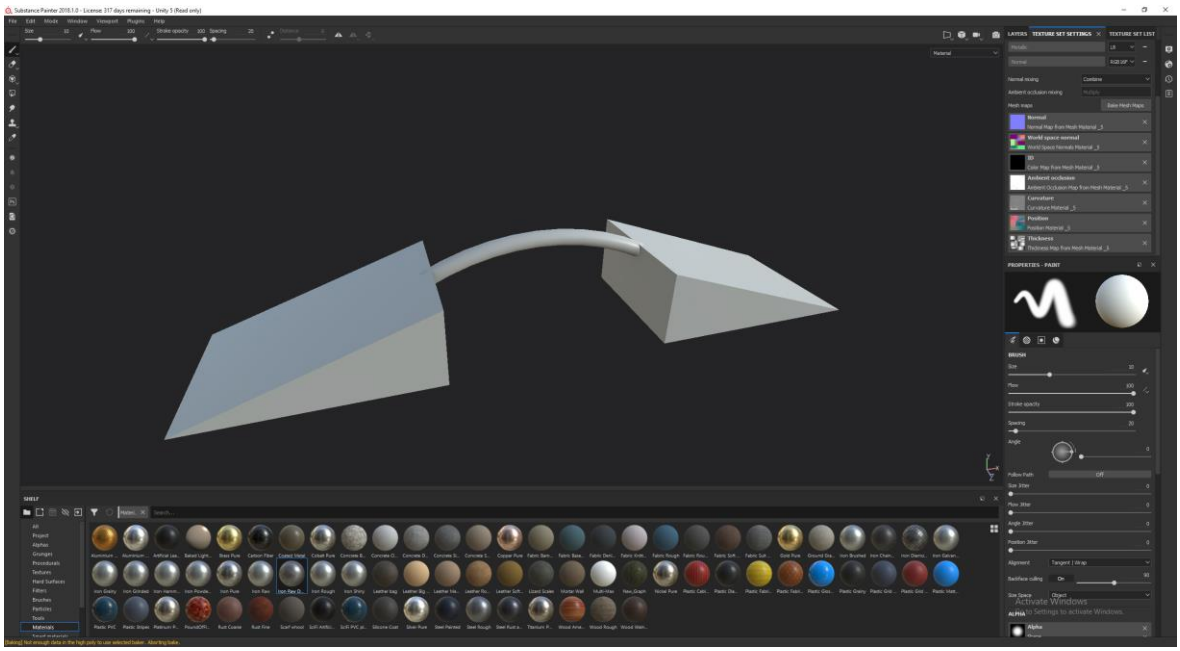
Properties je prozor u kojemu možemo dodatno podešavati opcije *Smart* maski ili pak kistova kojima crtamo teksture te utjecati na same parametre kao što su: *color*, *height*, *rough*, metal, *emission*… sve kako bismo dobili željeni rezultat naših tekstura.

3.2. Uvoz 3D objekta u *Substance painter*

Da bismo otvorili *FBX file*, odnosno naš 3D objekt izvezen i program za modeliranje, odlazimo na izbornik u *Substance painteru* na *File -> New*. Otvara nam se skočni prozor na kojemu odabirem *template „Unity 5*“, što je zapravo program kojim ćemo se korstiti u sljedećem koraku za izradu samog alata. Pod opcijom *Mesh* izabiremo naš 3D objekt koji želimo teksturirati. Također izabiremo rezoluciju koja će biti naša tekstura, ja sam korsitio rezoluciju 2048 x 2048 piksela, nije nužno defeinirati rezolucije u ovom koraku, prije exporta teksutra možemo veličinu naknadno podeiti. Klikom na gumb OK naš 3D objekt

pojavljuje se u sceni. Sljedeći je korak je „Baking“. Odlazimo na *Texture Set Settings* i stisnemo na gumb *Bake Mesh Maps*. Otvara nam se skočni prozor gdje možemo modifikacijom parametara za svaku mapu doći do rezultata koji želimo. Ja se koristim predefiniranim postavkama i klikom na gumb *Bake* započinje proces izrade mapa od 3D objekta.

Ako smo u prethodnom koraku modeliranja dodjeljivali ID indetifikatore objektu kako bismo razložili materijale, *Bake* je potrebno napraviti za svaki *Texture set*. Za sve dijelove svih objekata rabim isti materijal zato što postoji opcija maskiranja u *Substance painteru* gdje na brži način mogu razložiti jednostavne 3D objekte. Naš je objekt spreman za teksturiranje.



Slika 3-1. Uvezeni 3D objekt spreman za teksturiranje

3.3. Podjela materijala na 3D modelu

Kada na 3D objektu imamo više različitih materijala koje želimo naglasiti i teksturirati, te materijale možemo razložiti na dva načina: u programu za modeliranje dodijelivši 3D objektu, tj. skupini poligona određeni identifikator ili u programu *Substance painter* maskiranjem određenih slojeva 3D objekta dodjeljivanjem određenim objektima ili poligonima masku *layer* na kojoj radimo, odlučio sam se za soluciju unutar *Substance paintera*. Kao primjer uzimam 3D model elemnta koji se zove *Kicker + rainbow rail*, element je načinjen od metalne štange i 2 *kicker*-elemnta koji su izrađeni od betona. Stoga

imam 2 vrste materijala koje moram teksturirati. Da bih to postigao, kreiram 2 *Fill layera* koje sadržavaju masku na sebi. Prvo maskiram metalni dio na *Layeru 1*, a zatim maskiram betonski dio na *Layeru 2*. Sada je naš model spreman za teksturiranje.

3.4. Apliciranje tekstura na 3D model

Kada smo odredili *Fill layere* i maske, sljedeći korak je pronalazak odgovarajućih tekstura za naš objekt. Teksture kojima sam se koristio za *kicker*-elemente jednostavni je beton koji na sebi sadržava visok postotak *Roughnesa*, što mu daje svojstva da previše ne reflektira svjetlost previše kao što je to u stvarnome svijetu. Također sam rabio lagani *Height* efekt na objektu kako bih mu dao „nesvršenosti“ koje se tiču reljefa objekta, jer nijedan objekt u stvarnosti nije savršeno gladak.

Za šipku sam rabio teksture metala s vrlo malim postotkom *Heighta* i *Roughnessa* na nekim dijelovima kako bih također dobio efekt nesavršenosti, što na kraju rezultira većim fotorealizmom.

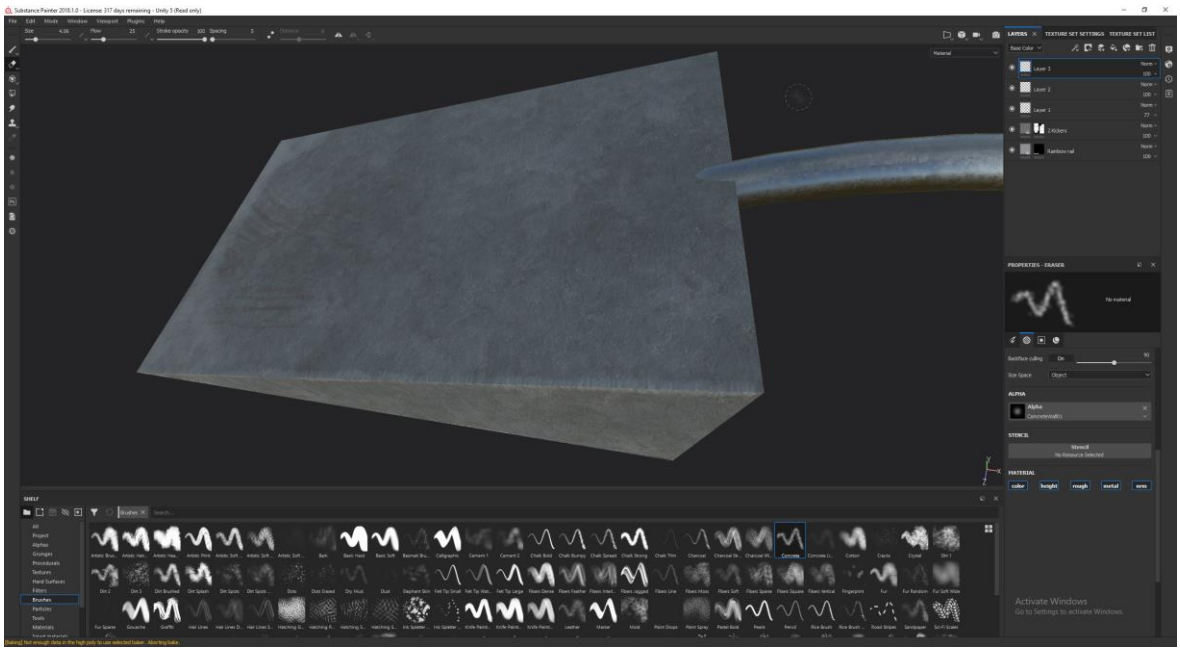
Teksture na objektu su zadovoljavajuće i spremne za *export*, ali ja sam se odlučio za još više dodavanja detalja s pomoću kistova kako bih postibao još veći efekt fotorealizma na objektu i teksturama.

Slika 3-2. Teksturirani 3D model *skate*-elementa u programu *Substance painter*

3.5. Dodavanje finih detalja na teksture

Ovisno o tome kako ćemo se koristiti našim 3D objektom s teksturama, moramo promisliti o tome želimo li mu dati dati više ili manje detalja. Ako u nekom slučaju naš model bude blizu kameri, bilo bi dobro da dodamo više detalja kako bismo ostvarili što bolji vizualni efekt. Ako je pak objekt daleko u pozadini scene i svrha mu je samo da upotpuni prostor, nije potrebno toliko raditi na detaljima jer oni neće biti ni uočljivi.

U slučaju mojih modela, oni će biti korišteni unutar alata za gradnju *skate*-parkova te će biti prezentirani na *web*-stranici kako bi posjetitelji imali što bolju predodžbu o s kakvim modelima rade. Stoga bi moji modeli trebali biti detaljnije teksturirani.



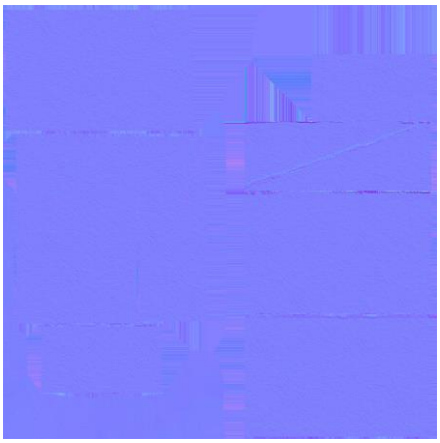
Slika 3-3. Rad na finim detaljima

3.6. *Export* tekstura za daljnji proces

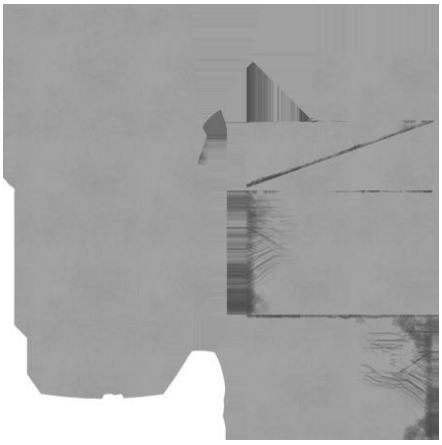
Kada smo zadovoljni izrađenim teksturama, odlazimo na opciju eksport koja se nalazi na *File - > Export textures...* Otvara nam se skočni prozor pod nazivom *Export document* te unutar njega možemo definirati na koju lokaciju želimo spremiti teksture, za koji program želimo spremiti teksture te u kojoj ih rezoluciji želimo spremiti. Lokaciju odabiremo kao poseban folder naziva tog modela kako bi teksture ostale organizirane. Stoga izabiremo

opciju *Unity 5* pod odabirom programa gdje mislimo rabiti te teksture i na kraju, ovisno o tome koju kvalitetu tekstura želimo, odabiremo rezoluciju samih tekstura. Za sve elemente rabio sam rezoluciju od 2048 x 2048 piksela. Rezolucija tekstura također se može posebno podešavati u programu u kojem izrađujem svoj alat.

Slika 3-4. Albedo Transparency mapa



Slika 3-5 Normal mapa



Slika 3-6. Rough mapa

4. Prezentacija 3D modela

3D modeli *skate*-elemenata sa teksturama bit će prezentirani unutar alata i na *webu* koji će sadržavati link za skidanje alata i galeriju slika tako da korisnik može vidjeti kojim modelima raspolaže prije negoli skine alat.

4.1. Renderiranje teksturiranih elemenata

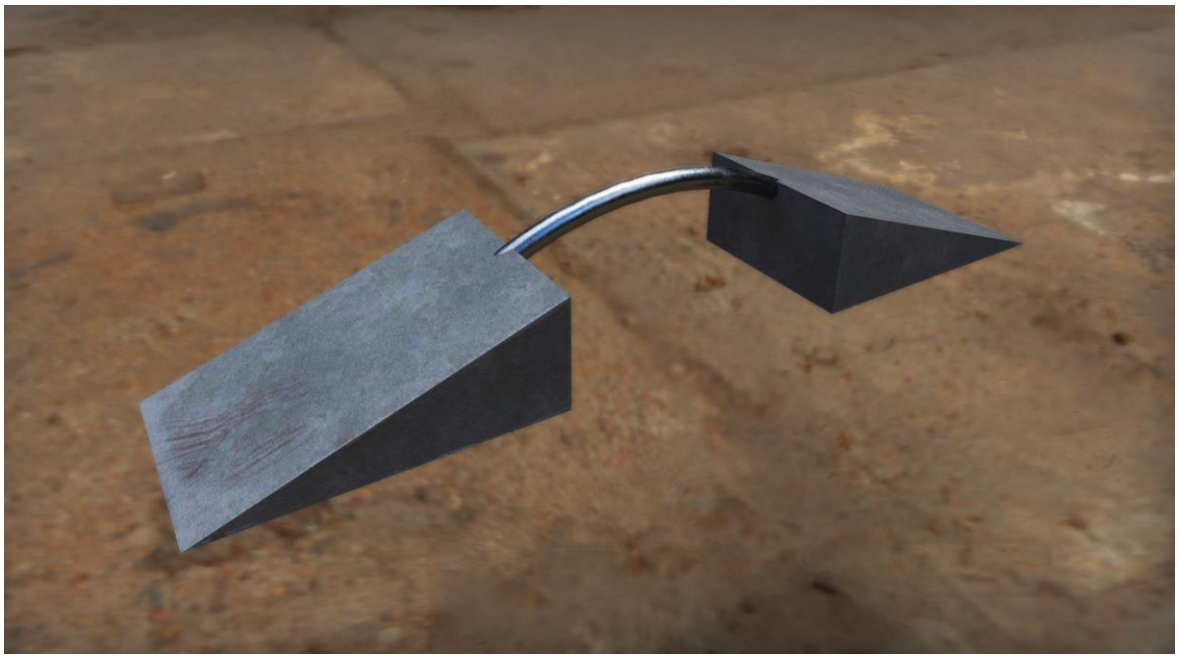
Slika 4-1 Primjer rendera teksturiranog 3D modela s HDRI pozadinom

Da bi slike kojima ću se koristiti na *webu* bile što bolje kvalitete, uporabljujem renderer koji se nalazi u *Substance painteru* (Iray) i s pomoću njega izrađujem slike visoke kvalitete.

Kao pozadinu slike rabim HDRI mape kako bih svoj 3D model smjestio u realni prostor i tako postigao još bolji vizualni efekt, a pokatkad i uočio neke greške koje ne uočavam kad okoliš nije aktiviran. Ako nisam zadovoljan dobivenim rezultatom, sliku još mogu popraviti u programu za rastersku grafiku.

4.2. Postprodukcija renderiranog materijala u programu *Adobe Photoshop*

Sliku naknadno možemo prepraviti u jednom od programa za rastersku grafiku, a ja sam izabrao program *Adobe Photoshop*. S pomoću njega prepravio sam neke stvari koje mi se nisu svidjele na slici. Prva stvar koju sam učinio bilo je centriranje objekta na sliku. To sam učinio tako da sam napravio *Crop* gornjeg dijela slike dok moj objekt nije došao u sredinu. Koristeći se maskama, zamaskirao sam okoliš oko objekta i prepravio im saturaciju da slika ima malo toplije boje. Malim prepravkama slika izgleda vizualno bolje te je spremna za *web*-stranicu.



Slika 4-2. Postprodukcija teksturiranog 3D modela

5. Izrada desktop aplikacije/alata za kreiranje dizajna *skate*-parkova u programu *Unity*

Kada su svi modele i teksture gotovi, možemo krenuti s izradom aplikacije za dizajniranje *skate*-parkova, no prije toga moramo definirati neke stvari kako bismo si olakšali posao. Aplikaciju ću izraditi s pomoću programa *Unity 5*. Taj program trenutačno podržava programski jezik C# i programski jezik *UnityScript*, poznat pod imenom „javascript za unity“. Potrebno je definirati i mehanike koje će se pojavljivati u aplikaciji. Također prije početka rada trebao bih imati spremne UI elemente koje će se rabiti unutar aplikacije. I na kraju potrebno je uvesti sve 3D modele *skate*-elemenata kojima ću se koristi u aplikaciji.

5.1. Definiranje programskog jezika za izradu aplikacije

Odlučio sam se za programski jezik C# jer na temelju istraživanja u *asset storeu* unutar *Unity* programa uočio sam da je većina materijala što se skripti tiče pisana u C# jeziku, a ja po svojim osobnim preferencijama ne bih želio kombinirati više jezika u jednom projektu radi što bolje organiziranosti.

S pomoću C# jezika pisao sam sve skripte kojima sam se koristio za:

- animiranje kamere
- promjenu između scena
- rotaciju i uništavanje objekta
- davanje funkcije UI elementu
- Drag & drop elemente
- zbrajanje cijena na podlozi na kojoj gradimo skejtpark
- pauza i *game* UI za nastavak i izlazak iz igre.

5.2. Definiranje mehanika za dizajniranje *skate*-parka

Mehanike za ovaj alat vrlo su jednostavne i podijelio sam ih u dvije skupine. Mehanike za interakciju s kamerom, mehanike koje daju mogućnost korisniku da interakcijom s UI elementima stvara 3D objekte *skate*-elementa koje razvrstava na danoj podlozi kako bi

napravio dizajn *skate*-parka i mehanika za uzimanje „Screen shota“ koja se aktivira na određeni pritisak slova na tipkovnici, te se zatim taj „Screen shot“, odnosno perspektiva dizajna sprema u folder u direktoriju gdje je instaliran sam alat.

Mehanike za interakciju s kamerom definiram kako bih imao uvid u ono što želim postići kao krajnji rezultat. Rabim jednu kameru u sceni kroz koju korisnik gleda na podlogu na kojoj izrađuje svoj dizajn. Kamera se može rotirati na rubove parka kako bi korisnik iz ptičje perspektive imao uvid u to kako njegov dizajn izgleda.

Stoga kamera sadržava 4 UI elementa „Button“ s pomoću koji dolazi to tog rezultata. Da bi korisnik što preciznije mogao slagati elemente također je potrebno unijesti UI element „Button“ koji ima funkciju da se korisnik prebaci u gornji pogled „Top view“. Korisnik također ima opciju da se vrati u svoj perspektivni pogled s pomoću UI elementa „Button“ pod nazivom *Perspective*. Ukupno 6 UI elemenata „Button“ služi za animiranje pogleda kamere na scenu. Kamera ima točku na koju se fokusira, ona se nalazi na sredini podloge na kojoj slažemo *skate*-elemente. Klikom na UI elemente vezane za kameru jedino manipuliramo njezinom pozicijom i rotacijom, a točka fokusa uvijek ostaje ista.

Mehanika za stvaranje 3D modela s pomoću UI elemenata mora postojati kako bi korisnik mogao stvoriti 3D objekt na sceni. Taj sam problem riješio tako što sam s lijeve strane scene kreirao izbornik koji sadržava UI elemente „Buttona“ dvadeset jednog *skate*-elementa koji su ponuđeni u alatu. Lijevim klikom miša na određeni „Button“ i njegovim povlačenjem na podlogu za izradu dizajna stvara se taj 3D objekt čija se slika nalazila na samom „Buttonu“.

Mehanika za manipulaciju 3D modela s pomoću miša. Kada imamo željeni *skate*-element na podlozi, pojavljuje se potreba da ga smjestimo na određeni dio *skate*-parka i rotiramo ga da bismo dobili željeni dizajn. Lijevim klikom miša na element i držanjem pomičemo taj element na određenu poziciju na podlozi i otpuštamo lijevi klik kako bismo ga tamo i ostavili. Također se pojavljuje potreba za rotacijom istog elementa da bismo složili elemente tako da budu funkcionalni u svim smjerovima, desni klik miša na element rotira taj isti element za 180 stupnjeva u intervalu od jedne sekunde. Ako se korisnik alata odluči da ipak ne želi određene elemente na podlozi, postoji funkcija za brisanje elementa srednjim klikom na taj element.

Mehanika za spremanje dizajna zadnji je korak u izradi dizajna. Korisniku je omogućeno spremanje slike dizajna kako bi ga poslije mogao dijeliti ili upotrebljavati u

svom poslovnom procesu. Korisnik može spremiti 5 različitih slika perspektiva istog dizajna pritiskom na slovo „P“ na tipkovnici, a te se slike zatim spremaju u folder „Screenshots“ koji je lociran u direktoriju u koji je instalirao igru.

Kada smo definirali sve željene mehanike, spremni smo krenuti s dizajnom UI elementa koji će dobiti navedene funkcije.

5.3. Dizajniranje UI elemenata aplikacije u programu *Adobe Photoshop*

Nakon definiranih mehanika imamo uvid u sve funkcije kojima ćemo se koristiti. Te funkcije moraju biti vizualno prisutne unutar alata kako bi ga korisnik mogao upotrebljavati, a funkcije poprimaju oblik UI elemenata „Button“ i stoga ih moramo napraviti vizualno atraktivnima i jednostavnima kako bi se korisnik mogao što lakše služiti njima.

Da bismo to postigli, moramo prvo napraviti njihov dizajn, a zatim ih izvesti iz programa da bismo ih mogli uvesti u program *Unity 5*, u kojih ćemo ih i implementirati.

Programi koje sam rabio za dizajn UI elementa jesu *Adobe Photoshop* i *Adobe Illustrator*.

Dizajn početnog prikaza jest dizajn koji će se prvi prikazati korisniku aplikacije kada se aplikacija otvori. Na tom zaslonu korisnik će dobiti poruku dobrodošlice. Dobit će grafički prikaz funkcija, upute kako da se koristi mišem i što sve njime može napraviti, dobiti će 3 savjeta kako napraviti bolji dizajn *skate*-parka i u sredini zaslona se nalazi UI element „Button“ koji vodi korisnika na sljedeću scenu.

Paletu boja koju upotrebljavam jest više nijansi zelene i žute jer će mi takvi biti ostali UI elementi i okoliš u aplikaciji pa stoga želim da boje ne budu u više od 2 tona.

Za prvi zaslon koristim se jednom vrstom tipografije.

Slika 5-1. Početni zaslon aplikacije

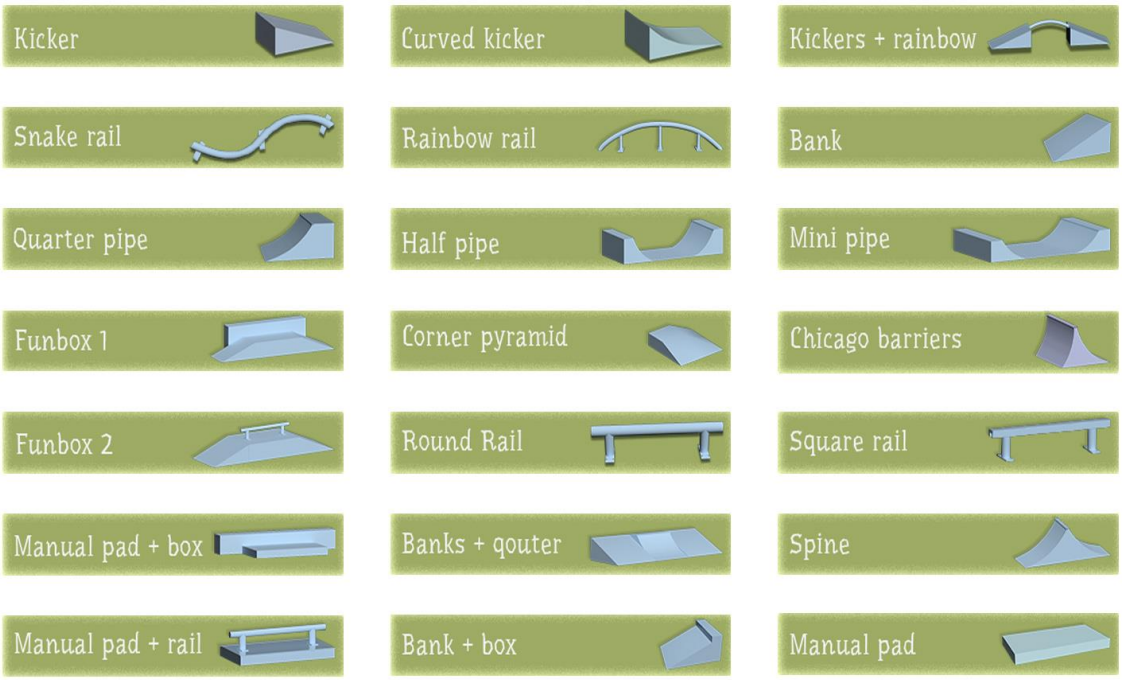
Font uporabljen u ovom dizajnu nije nimalo ozbiljan te podsjeća na stare dizajne grafika dasaka za *skateboard*. Samim fontom tako pripremam korisnika aplikacije na priču o *skateboardingu*.

Pozadina samog dizajna načinjena je od teksture betona koja je prekrivena žuto-zelenim gradijentom s određenim postotkom transparentnosti. Ovim potezom suptilno nagovještavam korisniku da je riječ o betonskim skejtparkovima i elementima, a gradijentom koji prekriva teksturu dajem mu na znanje da će se primjenjivati ta paleta boja.

Dizajn UI elemenata „Button“ kartica *skate*-elemenata radimo za „buttone“ koji se nalaze s lijeve strane prikaza. Interakcijom s tim karticama korisnik stvara 3D objekt na podlozi, gdje će nastati dizajn njegova *skate*-parka. Stoga te UI elemnte moramo vizualno predočiti korisniku kako bi znao koji će *skate*-element stvoriti u interakciji s „buttonom“.

Na tom sam „buttonu“ odlučio staviti tekst, odnosno naziv tog *skate*-elementa i sliku tog elementa kako bi korisnik vidio što zapravo stvara.

Paleta boja rabljena za UI elemente „button“ jest zelena s efektima žute koji su dodani naknadno. Font se razlikuje od stadarnih i glavnih UI elemneta da bih se istaknuo od ostatka elemenata na sceni



Slika 5-2. UI elementi „button“ rabljeni u aplikaciji za stvaranje *skate*-elemenata

Dizajn UI elemenata „button“ pogleda kamere. Kameri smo dali funkciju da se rotira na pritisak gumba pa potrebno je grafički stilizirati „buttone“ kako bi korisnik znao koja im je funkcija.

Za „buttone“ koji aktiviraju kameru rabio sam drukčiji dizajn kako bi ih korisnik drukčije percipirao. „Buttoni“ koriste različiti font od dosad primjenjivanih, ali ostaju unutar palete boja.

Slika 5-3. Dizajn UI elemenata „button“ pogleda kamere

Na „buttonima“ *Perspective* i *Top view* dodane su ikone kako bi korisniku „buttoni“ bili što vizualnije pojednostavnjeni.

Dizajn UI opisa mehankike za spremanje dizajna. Kada korisnik ima finalni raspored *skate*-elemenata na podlozi i kada je odabrao jedan o pogleda kamere na dizajn može stisnuti slovo „P“ kako bi spremio taj dizajn. Da bi korisnik znao za tu mehaniku, potreban je opis samog UI-ja. Taj sam problem sam riješio tako što sam dodao transparentnu sliku opisa kako to zapravo može i postići.

Slika 5-4. UI opis za spremanje dizajna parka

Opis je smješten u gornjem desnom kutu scene iznad dvaju „buttona“ koji služe za manipulaciju kamerom, a samim tim i taj opis dolazi do izražaja kada se korisnik koristi kamerom.

I zadnja stvar koja pripada UI elementima jest *Pause menu*.

On je načinjen od teksta i dvaju „buttona“, *Resume* i *Quit*, koji imaju funkciju vraćanja u aplikaciju ili izlazak iz nje.

Za dizajn je uporabljena jedna vrsta fonta u skladu s paletom boja koja je rabljena za druge UI elemente.

Slika 5-5. UI elemnti *Pause menu* sekcije

5.4. Izrada skripti u C# programskom jeziku

Sada kada imamo sve modele i UI elemente spremne za uporabu vrijeme je da im damo funkciju s pomoću jezika C# koji smo definirali na početku poglavlja.

Kronološkim redoslijedom prva skripta koja dolazi na red jest skripta koja na pritisak „buttona“ mijenja scenu aplikacije.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class LetsBuild : MonoBehaviour {
7
8      public void PlayGame ()
9      {
10
11         SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1);
12
13     }
14
15 }
16
```

Slika 5-6. LetsBuild.cs skripta za aktiviranje sljedeće scene

Ovo je primjer vrlo jednostavne skripte, a napisao sam je da bih klikom na gumb mogao pokrenuti sljedeću scenu.

Skripta funkcionira tako da s pomoću klase *SceneManager* pokrećem scenu koja je inkrementirana za 1 indeks, što znači da pokreće sljedeću scenu po redu. Scene također možemo pozvati unutar klase prema imenu.

Kada smo kliknli „button“ i ON skripta je odradila svoju funkciju, nalazimo se na sljedećoj i zadnjoj sceni koja se zove „Pocetak“, što je zapravo sama jezgra aplikacije, scena s UI „buttonima“, 3D elementima i funkcionalnošću za spremanje dizajna.

```
5  public class CameraMovment : MonoBehaviour {
6
7      public Transform TopCorner;
8      public Transform BottomCorner;
9      public Transform LeftCorner;
10     public Transform RightCorner;
11     public Transform MiddleLookAt;
12
13     public Transform TopView;
14     public Transform Perspective;
15
16     public Transform CurrentCorner;
17
18     public float speed;
19     // Use this for initialization
20
21     void Start ()
22     {
23
24     }
25
26     // Update is called once per frame
27     void Update ()
28     {
29         transform.position = Vector3.Lerp (transform.position, CurrentCorner.position, speed * Time.deltaTime);
30         transform.LookAt (MiddleLookAt.position);
31     }
32
33     public void TopCornerPressed()
34     {
35         CurrentCorner = TopCorner;
36     }
37
38     public void BottomCornerPressed()
39     {
40         CurrentCorner = BottomCorner;
41     }
42
43     public void LeftCornerPressed()
44     {
45         CurrentCorner = LeftCorner;
46     }
47
48     public void RightCornerPressed()
49     {
50         CurrentCorner = RightCorner;
51     }
52
53     public void TopViewPressed()
54     {
55         CurrentCorner = TopView;
56     }
57
58     public void PerspectivePressed()
59     {
60         CurrentCorner = Perspective;
61     }
62 }
```

Slika 5-7 CameraMovement.cs skripta za animiranje kamere na klik

CameraMovement.cs skripta funkcionira tako da se pomiče na predefinirane pozicije kutova *skate*-parka u određenom vremenskom intervalu.

Da bismo odredili te kutove, prvo moramo definirati 4 varijable: *TopCorner*, *BottomCorner*, *LeftCorner*, *RightCorner*. Nakon što imamo te 4 varijable, odlazimo u scene *view* i tamo kreiramo 4 nevidljiva *game*-objekta. Te *game*-objekte slažemo u 4 kuta na kojima želimo pozicionirati kameru klikom miša na „buttone“.

Također moramo deklarirati varijablu pod nazivom *MiddleLookAt* koja će sadržavati poziciju pogleda kamere. Za *MidleLookAt* poziciju nevidljivog *game*-objekta rabimo čistu sredinu same scene, odnosno podloge na koju slažemo *skate*-elemente (x = 0, y = 0, z = 0).

Također moram deklarirati varijablu *CurrentCorner* koja predočuje poziciju trenutačnog *game*-objekta na kojemu se nalazi kamera. Svakim klikom na drugi „button“ ta *CurrentCorner* postaje novi *game*-objekt.

Zadnja varijabla koju moramo deklarirati ako želimo dodati malo ljepšu interakciju jest varijabla *speed*. To je varijabla koja prima samo brojeve, i to samo cijele brojeve.

Kombinacijom svih tih varijabli unutar funkcije *void Update* dobivamo željeni rezultat animacije kamere. Funkcija *Update* poziva se svaki *frame* unutar aplikacije.

```
5 public class PauseMenu : MonoBehaviour {
6
7     public static bool GameIsPaused = false;
8
9     public GameObject pauseMenuUI;
10    // Update is called once per frame
11    void Update () {
12        if (Input.GetKeyDown (KeyCode.Escape)) {
13            if (GameIsPaused) {
14                Resume ();
15            } else {
16                Pause();
17            }
18        }
19    }
20
21    public void Resume (){
22        pauseMenuUI.SetActive (false);
23        Time.timeScale = 1f;
24        GameIsPaused = false;
25    }
26
27    void Pause(){
28        pauseMenuUI.SetActive (true);
29        Time.timeScale = 0f;
30        GameIsPaused = true;
31    }
32
33    public void QuitGame(){
34        Application.Quit ();
35    }
36 }
37
```

Slika 5-8 PauseMenu.cs skripta za pauziranje igre

Kako bih korisniku uljepšao *User expiriance* i dao opciju da izađe iz aplikacije klikom na gumb, odlučio sam izraditi *Pause menu* UI i dati mu funkcionalnost.

Skripta je također jednostavna i pritisak tipke *Escape* na tipkovnici pokreće *Canvas* objekt koji na sebi sadržava tekstualni naputak korisniku da prije izlaska iz aplikacije spremi svoj dizajn.

Sadržava 2 „buttona“, jedan ima funkcionalnost ponovnog vraćanja u scenu za gradnju koji se zove *Resume,* a drugi se „button“ zove *Quit* i služi za izlazak iz cijele aplikacije.

Trenutačno su gotove 3 skripte koje funkcioniraju te možemo krenuti s unosom 3D objekata u program.

5.5. Import izrađenih 3D *skate*-elmenata u program

Da bismo na sceni imali identičan 3D objekt koji smo izvezli iz programa za modeliranje te sa svim pripadajućim teksturama koje smo izvezli iz programa za teksturiranje, potrebno je proći nekoliko koraka.

Prvi je korak da sam objekt unosimo u program *Unity*. Uzimamo objekt lijevim klikom miša i vučemo ga u program *Unity* i ispuštamo u željeni folder. Svoje objekte spremao sam u folder *Assets -> Models*. FBX objekti imaju parametar unutar *Unityja* koji se zove *Scale factor*. *Scale faktor* na svim elementima želim zadržati isti jer on utječe na proporcije modela.

Nakon što su modeli uneseni, da bih aplicirao teksture na te elemente, prvo moram kreirati objekt *Material* unutar programa *Unity*. Pritiskom na desni klik odlazim na *Create -> material* i imam novi materijal na kojem mogu raditi.

Zanimaju me 3 parametra na koje apliciram teksture:

- Albedo
- Metallic
- Normal.

Albedo je parametar koji uzima stvarnu boju ili sliku koja će biti prikazana na modelu. Parametar *Metalic* uzima vrijednosti na kojim dijelovima teksture prihvaća refleksiju od izvora svjetlosti, a mapa *Normal* parametar je koji daje prividne deformacije 3D objekta. Kada podesim sve parametre, povezujem materijal s 3D modelom i spreman je za uporabu.

5.6. Aplikacija potrebnih skripti na 3D modele i UI elemente

Nakon importa 3D objekata i izrade materijala te importa 2D UI elemenata i napisanih standardnih skripti za rad ostaje nam da napišemo 3 skripte kako bismo dali funkcionalnost 3D objektima i implementiramo *plugin* za uzimanje *screenshot* dizajna. Poslije tih koraka aplikacija je završena.

Slika 5-9. 2D UI elementi na sceni

2D UI elemente složio sam na scenu tako da prostor u sredini scene ostane vidljiv za slaganje dizajna. S lijeve strane nalazi se 2D UI s karticama *skate*-elemenata koje predočuju svaki element, *drag&drop* potezom kartice na plohu za slaganje elemenata bit će kreiran 3D model *skate*-elementa kada mu damo tu funkciju preko skripti.

U sredini scene na vrhu nalaze se 2D UI „button“ elementi za manipulaciju kamerom koji aktivacijom šalju kameru u drugi kut te je rotiraju da gleda prema sredini scene. Upotrebljava se za pregled dizajna iz četiriju perspektivnih kutova.

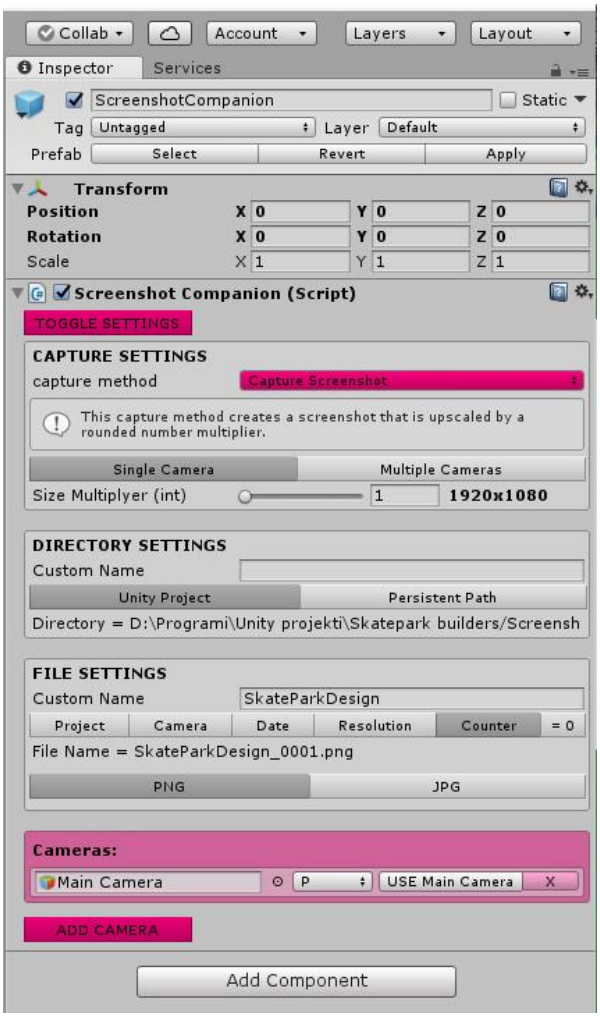
U gornjem desnom kutu nalazi se tekstualni opis kako spremiti dizajn. Pritiskom na slovo „P“ na tipkovnici dizajn iz aktivnog pogleda kamere sprema se u folder „Screenshots“, gdje je korisnik instalirao aplikaciju.

Ispod tekstualnog opisa nalaze se 2 2D UI „button“ elementa pod nazivom *Perspective* i *Top view* koji služe također za pomicanje kamere iz perspektivnog pogleda u *Top view* i iz *Top viewa* natrag u perspektivni pogled.

Ispod grupe kartica nalazi se tekstualni dio „Price“ koji zbraja sve vrijednosti *skate*-elemenata na podlozi te tako računa vrijednost samog *skate*-parka.

U *asset storeu* pronalazim besplatni dodatak kojim se mogu koristiti, a njegov je naziv „ScreenshotCompanion“. To je dodatak koji rabimo tako da kreiramo nevidljivi *game*-

objekt na sceni te mu dodjeljujemo skriptu „ScreenshotCompanion.cs“ kako bismo mogli konfigurirati kako želimo spremati naš dizajn.



Slika 5-10. Konfiguracija *ScreenshotCompanion plugina*

Prvo postavljamo poziciju nevidljivog *game*-objekta na sredinu određujući mu koordinate na osima X, Y, Z na 0.

Postoji više metoda za snimanje scene, a mi odabiremo „Capture Screenshot“ jer je to opcija da sprema istovjetnu sliku koju korisnik vidi u trenutku spremanja.

Imamo samo jednu kameru u cijeloj sceni pa stoga odabiremo snimanje samo jednom kamerom te određujemo da će slika koju spremamo biti rezolucije 1920 x 1080 piksela, što je još i danas najzastupljenija rezolucija kojom se ljudi koriste.

Također sami možemo predefinirati gdje se sprema sam dizajn unutra „Directory settingsa“, ja ostavljam predefinirane postavke tako da aplikacija automatski stvara folder u instaliramo direktoriju na kosirnikovu računalu.

Konfiguriram ime spremljenog dizajna tako da glasi „SkateParkDesign“ te mu dodajem opciju „Counter“ koja će svaki sljedeći dizajn inkrementirati za jedan kako ne bi došlo do spremanja slike preko slike. Stoga, ako imamo 4 spremljena dizajna, imena fajlova slike bit će: SkateParkDesign_0001.png, SkateParkDesign_0002.png, SkateParkDesign_0003.png, SkateParkDesign_0004.png.

Postoji opcija da izaberem hoću li spremati dizajn u PNG ili JPG formatu, a ja izabirem JPG format jer nemam transparentnosti na dizajnu.

I na samom kraju dodajem kameru kojom želim uzimati *Screenshoot*. to je moja jedina kamera na sceni „Main Camera“, a zatim postoji mogućnost dodati bilo koju tipku s tipkovnice koja će aktivirati skriptu. Ja sam izabrao slovo „P“.

Moj je *plugin* konfiguriran i spreman za uporabu, a pritiskom na tipku „P“ sprema dizajn u navedeni direktorij.

Sada mi ostaje davanje funkcije 2D UI karticama *skate*-elemenata i aplikacija skripti na 3D objekte kako bih ih mogao pomicati, rotirati i pobrisati sa scene.

Skripte na 3D objektu. Svaki 3D objekt *skate*-elementa na sebi ima 2 skripte pod nazivom „DragObjectScript.cs“ i „RotateAndDestroy.cs“.

Skripta „DragObjectScript.cs“ na sebi sadržava nekoliko parametara. Prvi je parametar „Concrete“. S pomoću tog parametra instanciram 3D objekt s podlogom na kojoj se nalazi taj 3D objekt.

Boolean parametrom pod nazivom „Mouse Over Me“ detektiram je li miš u interakciji s tim 3D modelom.

Sadržava i varijablu „Price“ gdje svakom objektu dodajem vrijednost kako bih poslije mogao sve zbrojiti i dobiti konačnu vrijednost *skate*-parka.

```
19      void Update (){
20
21          if (MouseOverMe && CS.DraggedPrefab == gameObject) {
22
23              RaycastHit hit;
24              Ray ray = Camera.main.ScreenPointToRay (Input.mousePosition);
25              if (Concrit.GetComponent<Collider> ().Raycast (ray, out hit, Mathf.Infinity)) {
26
27                  transform.position = new Vector3 (hit.point.x, hit.point.y , hit.point.z);
28
29              }
30
31          }
```

Slika 5-11. Dio skripte *DragObjectScript.cs* s funkcijom pomicanja objekta mišem

Skripta „DragObjectScript.cs“ funkcionira tako da kamera šalje nevidljive zrake prema objektu i tako detektira njegovu poziciji. Dok držimo lijevu tipku miša na objektu, ta se funkcija izvršava i svaki *frame* se računa nova pozicija tog objekta jer se taj dio koda nalazi u funkciji *Update*.

Skripta „RotateAndDestroy.cs“ sastoji se isto tako od nekoliko parametara. Prvi parametar je *speed*, to je *flooat* varijabla u koju možemo unositi cijele brojeve i tako usporiti ili ubrzati rotaciju. Ja sam svoju rotaciju postavio da se izvršava unutar jedne sekunde.

Sadržava još 4 glavna parametra koji detektiraju rotaciju tog objekta s pomoću *booleana*.

```
void Update (){

    if (mouseIsOver) {
        if (Input.GetMouseButtonDown(1))
        {
            rotationClicked = true;

        }

        if (Input.GetMouseButtonDown (2)) {
            gameObject.GetComponent<DragObjectScript> ().CS.totalPrice -= gameObject.GetComponent<DragObjectScript> ().objectPrice;
            Destroy (gameObject);
        }
    }
```

Slika 5-12. Jezgra skripte za rotaciju i brisanje

Na isječku ove skripte prikazana su dva uvjeta „if“. Prvi uvjet detektira je li kliknut desni klik koji se indeksira brojem (1) u jeziku C#. Ako jest, prosljeđuje tu informaciju u daljnji proces, gdje se objekt rotira za 90 stupnjeva na svaki klik unutar jedne sekunde.

Drugi uvjet „if“ detektira je li pritisnut srednji „button“ miša koji se u jeziku C# indeksira s brojem (2). Ako je kliknut, skripta izvršava naredbu koja briše vrijednost kliknutog objekta iz ukupne sume vrijednosti koja je definirana u „Price“ sekciji, kada se ta linija koda izvrši, pokreće se funkcionalnost brisanja objekta „Destroy (gameObject)“. Ta funkcija briše cijeli objekt sa svim skriptama iz scene.

Podloga na koju slažemo elemente povezana je sa skriptom koja računa kolika je ukupna vrijednost elemenata tako da zbraja vrijednost elemenata s ukupnom votom svaki put kada je element dodan na podlogu.

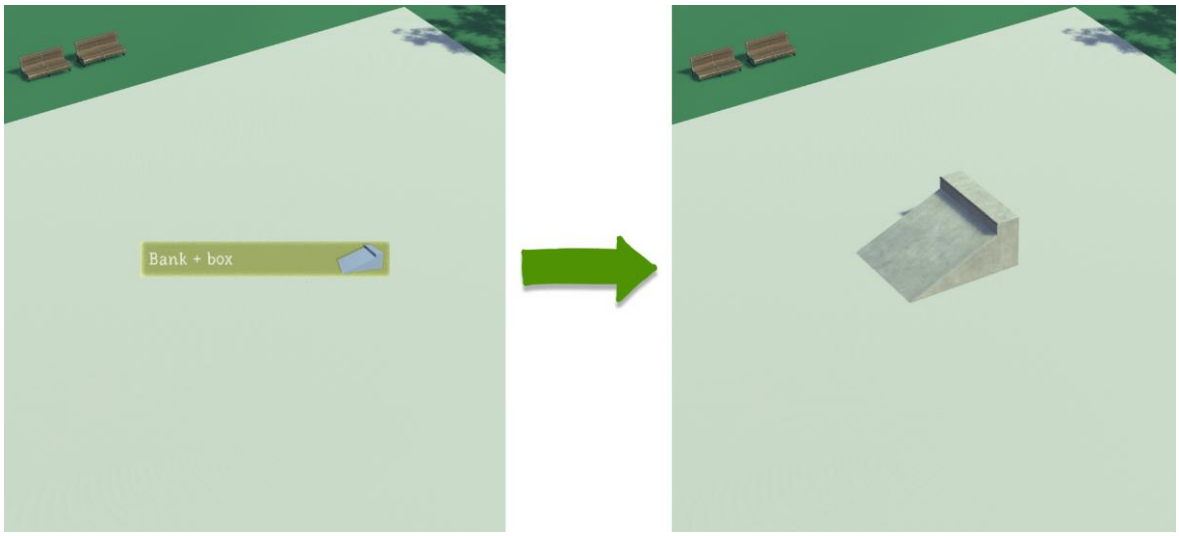
```
14      void Update (){
15          totalPriceText.text = "PRICE: " + totalPrice.ToString () + "$";
16      }
17
```

Slika 5-13. Dio skripte koja računa vrijednost svih elemenata

I za kraj, da bi ova aplikacija bila funkcionalna, trebam izraditi skriptu koja će 2D UI element pretvarati u 3D objekt na podlozi.

To sam učinio na sljedeći način. Napravio sam skriptu „cardScript.cs“. Prvo se traži *game*-objekt s Tag imenom „P“. „P“ sam postavio kao Tag ime podloge na koju slažem elemente. Taj se objekt sprema u varijablu pod nazivom „CS“. Zatim deklariram varijablu pod nazivm „preFab“ koja mi omogućuje da povežem odgovarajući model sa samom 2D UI karticom. Sada su model i kartica povezani.

Sljedeća stvar koju dodajem u skriptu jest funkcija da, kada korisnik povuče karticu lijevim klikom i ta se kartica detektira na podlozi na koju slažem elemente, na otpust lijevog klika miša na toj pozicije gdje se nalazi miš stvara se 3D objekt.



Slika 5-14. Primjer funkcioniranja skripte „cardScript.cs“

Aplikacija je spremna za izvoz iz programa *Unity 5* pa odbirem platformu za koju želim izvesti aplikaciju. Izabirem PC, Mac & Linux opciju te scene koje želim izvesti. U ovoj aplikaciji imam 2 scene te ih putem *check boxa* označujem.

Stisnem „button“ pod nazivom „Build“ i čekam da se moja aplikacija izveze u direktoriji koji sam odabrao. Pri pokretanju aplikacije pokazat će se dijalog gdje mogu odabrati rezoluciju u kojoj želim raditi. Ako želim, mogu dodati sliku koja će reprezentirati igru na dijalogu i mogu dodati posebnu ikonu koja će se prikazivati iznad file imena same datoteke.

5.7. Izrada *splash imagea* za pokretanje aplikacije

Splash image radim u programu *Adobe Photoshop*. Rezolucija slike koja će se prikazivati kao *splash image* u dijaloškom okviru je 512 x 256 piksela pa stoga pri kreiranju dokumenta izabirem tu rezoluciju.

Rabim *Screenshot* 3D objekta *skate*-elementa kako bi postavio kao pozadinu da korisnik već ima vizualni dojam s čim se susreće u aplikaciji.

Za tekstualni dio rabim stilizirani font koji sam upotrijebio na prvoj sceni pokretanja aplikacije i dodatno ga stiliziram efektima.

Iza samog teksta povlačim transparentni gradijent kako bi tekstualni dio došao više do izražaja.

Spremam dizajn slike i sada ga mogu rabiti unutar programa *Unity 5*.

Slika 5-15. *Splash image* korišten u dijalogu pri pokretanju aplikacije

5.8. Izrada ikone aplikacije

Za svaku aplikaciju ili igricu potrebno je napraviti ikonu kako bi se vizualno razlikovala od ostalih. Ikona bi trebala vizualno biti u srodstvu sa samim vizualima korištenima unutar aplikacije.

Stoga sam ja za ikonu svoje aplikacije poslužio jedan od 3D modela, odnosno *skate*-elemenata na zelenoj podlozi koja je izvučena iz palete boja same aplikacije.

Ikonu sam dizajnirao u većoj rezoluciji 500 x 500 kako bih je što preciznije mogao napraviti i posložiti, a zatim sam je spustio na rezoluciju 48 x 48 piksela, što i je stvarna rezolucija desktop ikone.

Slika 5-16. Dizajn desktop ikone za aplikaciju *Skatepark Builders*

5.9. *Export* aplikacije za desktop verziju

Nakon što unesemo sliku (*ikon*) i *splash image* u program *Unity* u naš projekt svi su aspekti aplikacije zadovoljeni i funkcionalni.

Odabiremo platformu za koju izvozimo aplikaciju i stisnemo na *izvoz*. Pričekamo da *Unity* sortira sve datoteke koje će se upotrebljavati u aplikaciji te, kada taj proces završi, naša je aplikacija završena i nalazi se u direktoriju koji smo odabrali prije izvoza.

Klikom na datoteku s nastavkom EXE pokrećemo aplikaciju.

6. Izrada *Set up* datoteke za instalaciju aplikacije

Da bih lakše dijelio i komprimirao datoteke koje se nalaze u mojoj aplikaciji, potrebno je napraviti instalacijsku datoteku.

Za kreiranje instalacijske datoteke rabio sam *Inno Setup software*. Taj besplatni softver omogućuje korisniku da u minimalnom broju koraka od svoje aplikacije kreiraju instalacijsku datoteku. Za moj slučaj ovo je bilo idealno rješenje.

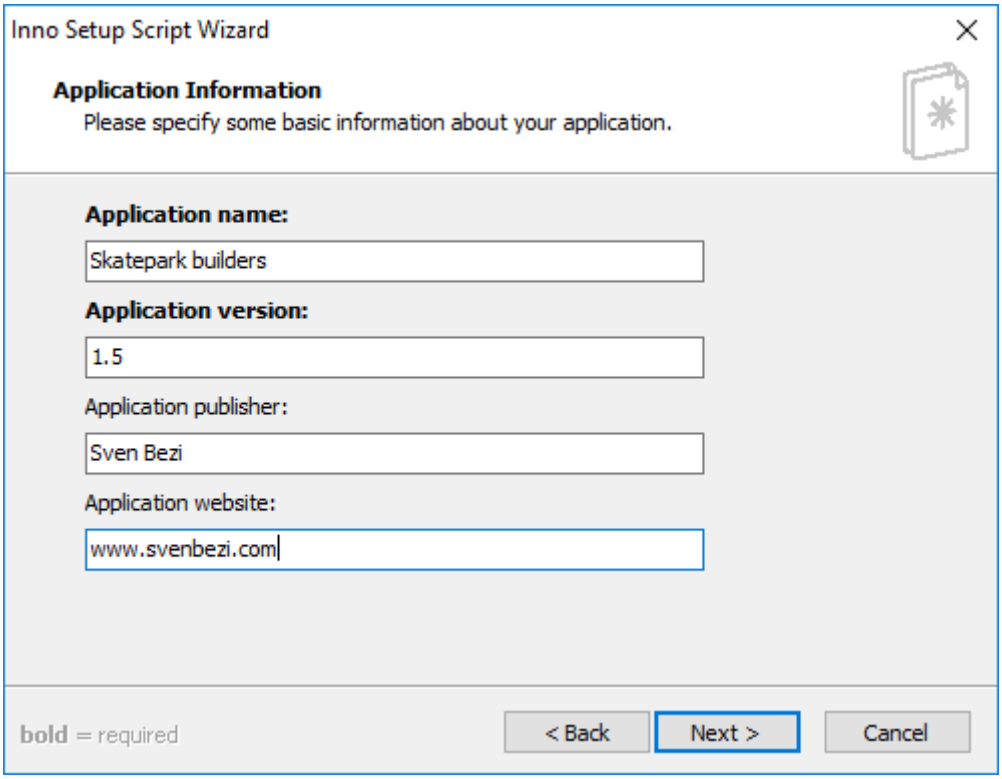
Inno Setup softwer u izradi instalacijske datoteke daje broj mogućnosti kako da konfiguriramo našu instalacijsku datoteku te samim time omogućimo korisniku slobodu da instalira aplikaciju po svojim preferencijama.

Mogućnosti koje nam pruža *Inno Setup software*:

- *Application name* – ime aplikacije koje će stajati u opisu *Set up* datoteke
- *Application version* – verzija aplikacije koja se pojavljuje u opisu
- *Application publisher* – ime izdavača aplikacije
- *Application website* – *web*-stranica aplikacije ili izdavača koja se nalazi u opisu aplikacije
- *Application destination base folder* – definiramo direktorij koji će biti ponuđen korisniku u kojemu se ta aplikacija instalira
- *Application folder name* – ime samog direktorija u kojem će biti instalirana aplikacija na definiranoj putanji
- *Ckeckbox: Allow user to change the application folder* – mogućnost da damo korisniku da mijenja instalacijski folder aplikacije
- *Application main executable file* – lokacija naše aplikacije odnosno exe datoteke
- *Other application files* – dodavanje svih ostalih potrebnih datoteka
- *Checkbox: Create a shortcut to the main executable in the common Start Menu Programs folder* – dodavanje prečaca na *start menu* izbornik
- *Checkbox: Allow user to create desktop shortcut* – davanje opcije korisniku da kreira prečac do aplikacije na desktopu
- *License file* – dodavanje tekstualne datoteke koja sadržava informacije o licenci

- *Inforamtion file shown before installation* – informacijska tekstualna datoteka koja sadržava određene informacije prije instalacije
- *Inforamtion file shown after installation* – informacijska tekstualna datoteka koja sadržava određene informacije poslije instalacije
- *Languages* – opcija za dodavanje drugih stranih jezika za instalaciju
- *Custom compiler output folder* – destinacije gdje će se generirati naša instalacijska datoteka
- *Compiler output base file name* – unos imena ime naše instalacijske datoteke
- *Custom Setup icon file* – mogućnost da dodajemo sami ikonu koja će se pojavljivati na našoj instalacijskoj datoteci
- *Setup password* – opcija da zaključamo instaler lozinkom koju odaberemo.

Kada smo popunili sve podatke i konfigurirali da se naša instalacijska datoteka ponaša kako mi želimo, završavamo cijeli proces klikom na gumb „Finish“ na zadnjem koraku i pričekamo da nam softver generira instalacijsku datoteku.



Slika 6-1. *Inno Setup software* za izradu instalacijske datoteke

7. Izrada prezentacijske *web*-stranice alata za dizajniranje *skate*-parkova

Proces izrade *web*-stranice sastoji se od 5 faza projekta. Tih 5 faza pokrivaju pitanja na koje moramo odgovoriti kako bismo na kraju procesa imali funkcionalnu *web*-stranicu. Te faze jesu:

- The Surface Plane
- The Skeleton Plane
- The Structure Plane
- The Scope Plane
- The Strategy plane.

Proces izrade kreće od *Strategy plane* te završava *Surface planeom*.

The strategy plane – Strategija uključuje istraživanje onoga što korisnici žele pronaći na *webu* te što im mi želimo pržiti putem *weba*.

The scope plane – To je opseg projekta koji definira od čega se sve elementi i funkcionalnosti sastoje. Opseg odgovara na pitanje hoće li neka funkcionalnost biti predviđena na *webu* ili ne.

The structure plane – To je nacrt *weba* i definira kako korisnik dolazi do određenih informacija, a struktura *weba* definira način na koji različiti elementi i funkcije djeluju zajedno.

The skeleton plane – Prije samog dizajna potrebno je napraviti kostur stranica, odnosno *wireframe*. Na *wireframeu* su određena mjesta za „tabove“, fotografije, blokovi teksta, gumbi...

The surface plane – jest sam vizualni dizajn *weba* koji uključuje slike, tekst, funkcionalnost i „klikabilnost“.

7.1. Strategijski plan

U strategijskome planu moramo analizirati i konstatirati potrebe korisnika koji će rabiti tu *web*-stranicu / aplikaciju i odrediti koji su joj ciljevi te na temelju tog istraživanja graditi temelje za našu *web*-stranicu.

7.1.1. Potrebe korisnika

U svakome *web*-projektu imamo ciljanu publiku kojoj želimo prezentirati sadržaj na našoj *web*-stranici, a prvi korak koji moramo napraviti jest segmentacija korisnika.

Segmentacija korisnika – Korisnike možemo segmentirati kroz brojne podjele, što više podjela uzmemo u obzir, naša će segmentacija biti preciznija. Neke od segmentacija kojima sam se koristio u svojem istraživanju jesu:

- zemljopisna segmentacija
- demografska segmentacija
- psihografska segmentacija.

Na temelju spomenutih segmentacija i prikupljenih podataka imam temelje kako bih izgradio osobe (persone) koje će se koristiti mojim *webom*.

Uporabljivost i korisnička istraživanja – Budući da je moj *web* reprezentativna stranica za aplikaciju, ne zahtijeva da posjeduje previše sadržaja. Odlučio sam da će moja stranica biti „one pager“ koji će biti podijeljen u 5 sekcija koje su *full screen* radi ljepšega vizualnog dojma.

Proveo sam istraživanje u kojemu sam korisnicima ponudio kartice s imenima sekcija da ih slože po svojoj intuitivnosti te redoslijedom odozgo prema dolje kako bi najlakše koristili stranici.

Kartice koje su bile ponuđene:

- kontakt kartica
- *Download* aplikacije kartica
- slike elemenata za gradnju parka kartica
- informacije za korištenje aplikacijama kartica
- video *preview* aplikacije kartica.

Analizom rezultata ispitanika došao sam do sljedećih rezultata. Sekcije na *web*-stranici trebale bi biti posložene sljedećim redoslijedom:

- video *preview* aplikacije kartica
- informacije za korištenje aplikacijama kartica
- *Download* aplikacije kartica
- slike elemenata za gradnju parka kartica.

Slika 7-1. Kronološki posložene kartice na temelju istraživanja korisnika

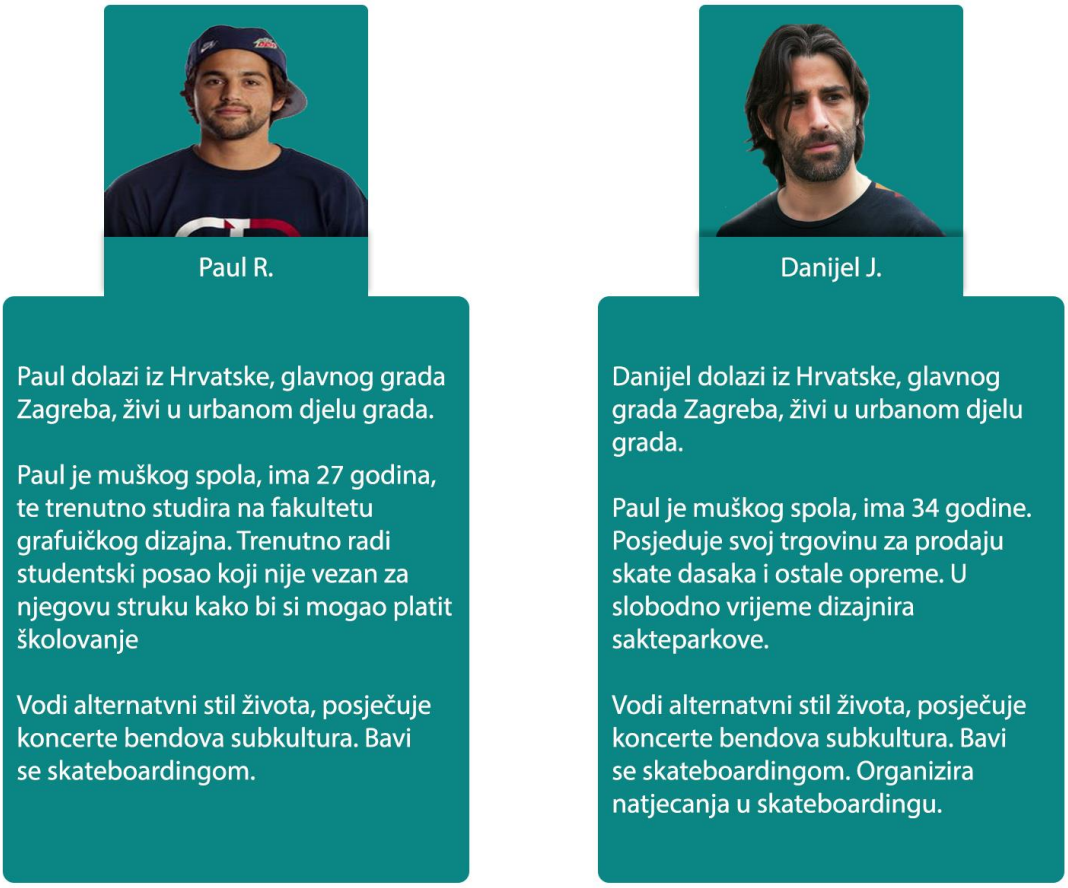
Kreiranje persone – Korisnike pretvaram u osobe sa sličnim osobinama koje sam izvukao iz segmentacije korisnika na temelju istraživanja zemljopisne segmentacije, demografske segmentacije i psihografske segmentacije.

Stoga krećem s izradom zamišljenog karaktera da predstavi potrebe cijele ciljane grupe korisnika.

Kreirao sam 2 persone.

Persona jedan je skejter.

Persona dva je dizajner skejtparkova.



Slika 7-2, Persone (osobe) koje će se koristiti stranicom i aplikacijom

7.1.2. Ciljevi proizvoda

Također trebamo definirati i ciljeve proizvoda, odnosno naše aplikacije i *web*-stranice, postavljajući si pitanje što očekujemo na kraju.

Trebamo definirati ciljeve našeg poslovanja, identitete našega brenda te pokazatelje uspješnosti našeg *weba*, odnosno koliko nam je *web* pomogao da ostvarimo svoj cilj.

Cilj poslovanja – Kao cilj poslovanja u većini slučajeva postavlja se „Ostvarivanje profita“ ili „Ušteda novca“.

Cilj poslovanja ovoga *web*-projekta zapravo je podijeliti funkcionalnu aplikaciju s ciljanim skupinama. Ovakvim potezom ne dobivam novčanu naknadu, ali gradim reputaciju. Aplikacija još ima vrlo mnogo prostora da se unaprijedi što planiram i napraviti u budućnosti. Na temelju ovakve aplikacije mogu pokrenuti *crowfounding* kampanju kako

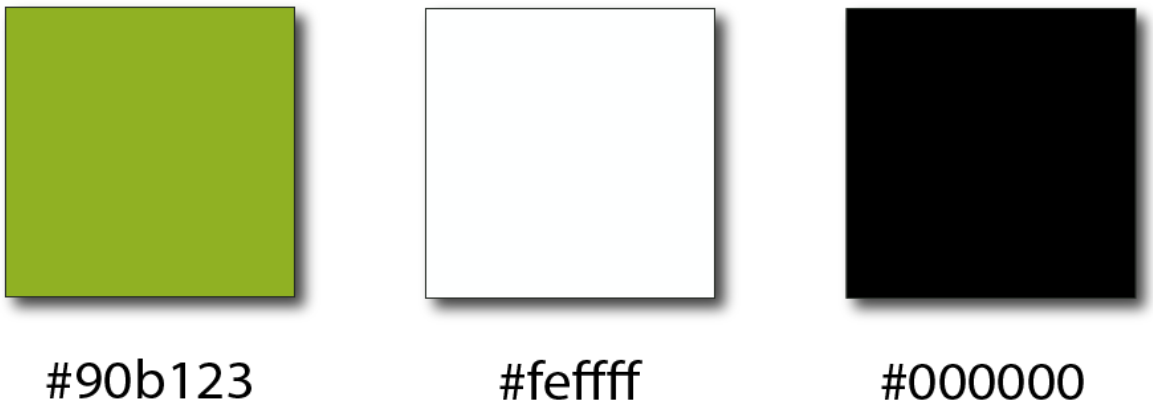
bih si osigurao resurse da mogu nastaviti raditi na toj aplikaciji i dodati joj još brojne mogućnosti, odnosno unaprijediti *User Expirience* i *User interface*.

Identitet brenda – Prije samog početka izrade također je potrebno odrediti *branding* naše *web*-stranice, odnosno stvoriti vizualni dojam i prezentaciju tvrtke, projekta ili proizvoda. odnosno one stvari zbog koje kreiramo *web*.

Svaki brend mora sadržavati svoj logotip, paletu boja i tipografiju kako bi kombinacijom svega toga stvorio prepoznatljivu sliku o sebi.

Logotip sam izradio u programu *Adobe Illustrator*, a logotip se sastoji od dvaju djelova, slike i teksta. Slika predočuje maskotu logotipa, a tekst naziv aplikacije.

Paleta boja uporabljena za izradu *weba* sastoji se od triju boja koje reprezentiraju brend i podsjećaju na boje uporabljene u aplikaciji.



Slika 7-3. Paleta boja uporabljena na *web-u*

Pokazatelj uspješnosti – To su metrike kojima mjerimo reakcije korisnika na naš projekt, odnosno proizvod kada je pušten na tržište.

Metrike kojima sam se koristio za svoj *web* pokazuju koliko puta treba kliknuti „button“ za *download* aplikacije.

Korsitim se također metrikama da vidim koliko vremena korisnik provede na stranici.

Te metrike pratim preko ugrađenog alata pod nazivom *Google Analytics*.

7.2. Plan opsega

Kada smo definirali što želimo i što žele naši korisnici, trebamo definirati kako zadovoljiti sve strateške ciljeve. Strategija postaje *scope* kada ono što korisnik želi i ciljeve projekta

pretvorimo u specifične i točno definirane zahtjeve u vezi s tim kakav sadržaj i funkcije želimo ponuditi korisniku da bismo ostvarili te ciljeve.

7.2.1. Funkcijske specifikacije

Funkcijske specifikacije označuju sve funkcije koje naš *web* treba imati. Obično se popisuju dvije vrste funkcijskih specifikacija. Zahtjevi na početku projekta i popis stvarnih funkcija na kraju projekta.

Kako je riječ o vrsti stranice „One pager“ podijeljenoj u 5 *fullscreen* sekcija, krećem od prve sekcije i opisujem funkcije za svaku.

Popis na kraju projekta mog *weba Skatepark Builders*

- Sekcija 1: Pozadina koja emitira video, video se emitira čim se stranica otvori. Preko videa prelazi tekst koji pozdravlja korisnika koji je došao na stranicu. Video prestaje s emitiranjem kada se korisnik krene navigirati prema sljedećoj sekciji.
- Sekcija 2: Kada korisnik dođe do pola sekcije 2, pojavljuje se tekst koji se stvara animacijom unutar jedne sekunde. Zatim se, nakon što je ta animacija gotova, također animacijom od 1 sekunde pojavljuju 3 paragrafa sa slikom kojom se opisuje kako se koristiti aplikacijom.
- Sekcija 3: Sadržava 3 elementa: tekst, sliku i „button“ koji se pojavljuju unutar jedne sekunde kada korisnik došao te sekcije. „Button“ ima funkciju da na klik krene skidati instalacijsku datoteku Skatepark Builders.
- Sekcija 4: Kada se korisnik do scrolla do sekcije 4 animacijom, pojavljuje se naslov sekcije; nakon što animacija završi, započinje sljedeća animacija koja stvara galeriju slika od 21 elementa kojim sam se koristio u aplikaciji. Klikom na *thumbnail* sliku da se otvara se *Lightbox galleria* slika koja ima funkcionalnostkoja omogučava koirsniku da lista slike sa strelicama na tipkovnici.
- Sekcija 5: Kada korisnik do *scrolla* do sekcije 5 pojavljuje se animirani naslov, animirane natuknice kako koristiti kontakt form i kontakt form. Sve animacije traju 1 sekundu. Kontakt *form* ima opciju upisivanja imena, teme, i poruke, te opciju za dodavanje slike koja će biti poslana na moj mail klikom na gumb *send*.

7.2.2. Zahtjevi za sadržajem

U zahtjevima za sadržaj potrebno je popisati sav sadržaj koji će se nalaziti na našoj stranici.

Taj sadržaj može biti u tekstualnom, video, audio ili grafičkom obliku.

Sadržaji kojima sam se koristio za svoj *web* jesu sljedeći.

- *Background* video za prvu sekciju koji prikazuje snimljeni dio iz aplikacije
- tekstualna grafika koja pozdravlja posjetitelja stranice
- tekstualne informacije koje educiraju posjetitelja kako se koristiti aplikacijom koju skida
- grafika sa strelicom koja pokazuje gdje se nalazi *download button*
- *Background image* koji sadržava tekst i grafiku „buttona“ za skidanje aplikacije
- 21 slika *skate*-elemenata koje će korisnik moći pregledavati unutar interaktivne galerije
- tekstualni dio koji opisuje kako rabiti *kontakt form*
- *Background image* za *button* u kontakt formu koji omogućuje *upload* slike.

7.3. Strukturni plan

Kada smo definirali strategiju, te kada imamo popis svih funkcijskih specifikacija koje ulaze u projekt i svih sadržaja propisanih možemo krenuti s izgradnjom strukture naše *web*- stranice.

Pri definiranju strukture potrebno je obratiti pozornost na dizajn interakcije i arhitekturu informacija.

Ako zadovoljimo kriterije u objema stvarima, imamo funkcionalnu strukturu *web*-stranice.

7.3.1. Dizajn interakcije

Opisuje moguća ponašanja korisnika i definira kako će sustav odgovarati na to ponašanje. Programeri su fokusirani na dvije stvari: što radi i kako radi. Tehnički je ispravan, ali nije najbolje rješenje za korisnika. Što sustav radi kada korisnik napravi grešku i što sustav može napraviti da se takve greške ne događaju?

Moj slučaj „One pagera“ ne daje korisniku preveliku slobodu i opcije zato što ne postoji mnogo scenarija kako se korisnik može koristiti stranicom.

Stranica je podijeljena u 5 sekcija. Na stranici ne postoji standardna navigacija s linkovima, nego sam se za rješenje navigacije koristio opcijom „Dot navigation“ koja se nalazi sa strane kao mali sidebar i sadržava pet točaka na sebi. Svaka točka ima funkciju da scrolla do navedene sekcije.

Korisnik se ne može izgubiti na stranici jer je osuđen na jedan HTML dokument po kojem jedino može „skrolati“ gore-dolje.

Postoje još tri interaktivna dijela na stranici.

Sljedeća je interaktivnost *download button*. Klikom na njega korisnik skida aplikaciju koja se nalazi na našem *webu*. *Button* je velik, jasan i uočljiv te reprezentira svoju funkciju vizualno.

Iduća interaktivnost jest galerija slika. Slike su složene u 7 stupaca po 3 slike, *hover over* slike aktivira animaciju koja daje korisniku na znanje da je slika interaktivna. Klikom na sliku otvara se *full screen* galerija slika koju korisnik može listati lijevo-desno strelicama na tipkovnici.

Zadnja interaktivnost na stranici jest kontakt *form* koji nakon upisanih podataka šalje na *mail* koji sam definirao. U ovom slučaju koristio sam se svojom osobnom e-poštom.

Kontakt *form* im dodatnu interaktivnost koja korisniku omogućuje da na *drag & drop* „uplouda“ svoje slike te i njih pošalje putem forme.

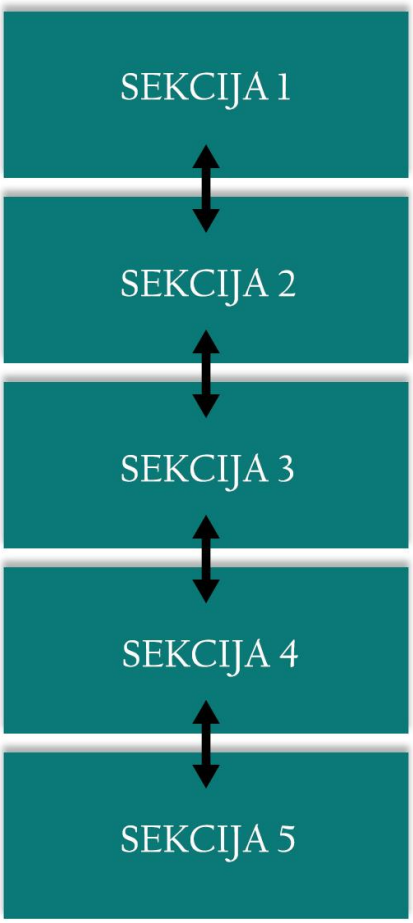
7.3.2. Arhitektura informacija

Što korisnici očekuju od informacija – gdje ih pronaći? Kreiranje organizacijskih i navigacijskih shema omogućuju kretanje korisnika kroz *web*.

Structuring content označuje dizajniranje sustava koji omogućuje korisniku da jednostavno dođe do informacija.

Korisnik na mojoj stranici sve informacije nalazi pod jednom HTML datotekom, odnosno jednom stranicom na kojoj su sve informacije.

Stranica posjeduje 5 *full screen* sekcija koja nudi sadržaj korisniku ili ga informira o samoj aplikaciji. Korisnik s 1. *screena* u jednom kliku može doći do zadnjeg.



Slika 7-4. Ruta navigacije za dolazak do informacija

7.4. *Wireframe* stranice

Detaljnije razvijamo strukturu, identificiramo specifične zahtjeve dizajna, navigacije i prezentacije informacija koje će strukturu učiniti konkretnom nazivamo *wireframe* stranicama.

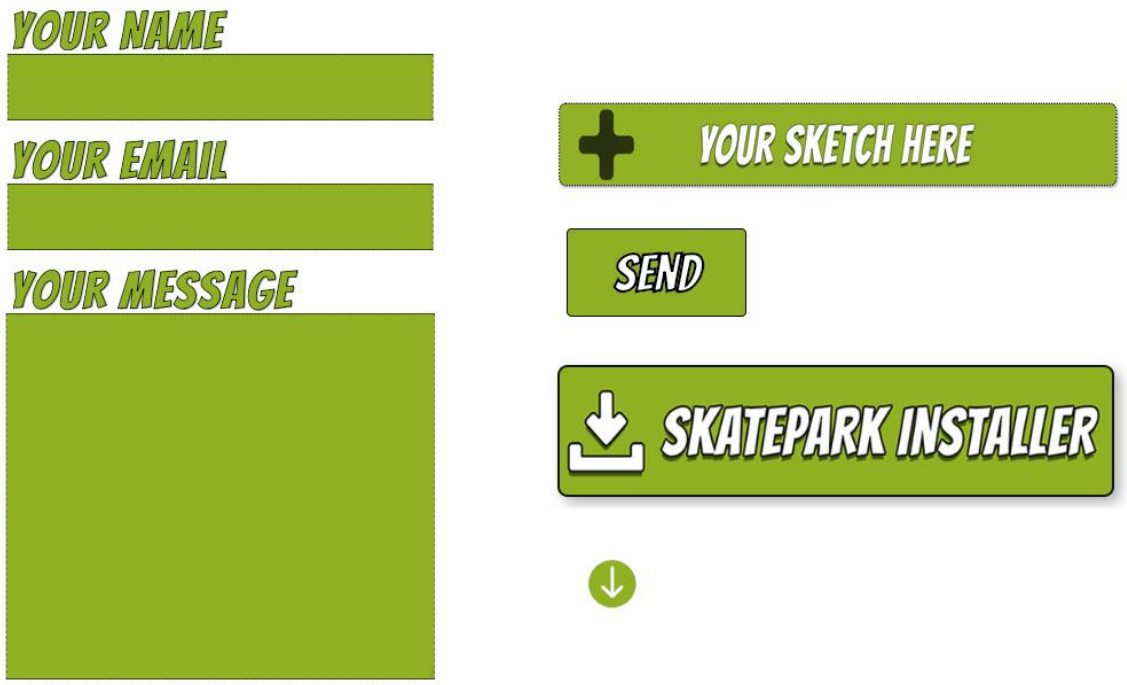
U ovom koraku razvijamo individualne komponente i njihov odnos s ostatkom stranice.

Da bismo napravili kvalitetne *wireframe* stranice moramo obraditi 3 teme, a to su:

- korisničko sučelje
- dizajn navigacije
- dizajn informacija.

7.4.1. Korisničko sučelje

Dobro korisničko sučelje ključ je uspjeha *web*-stranice. Važno je odabrati odgovarajući element korisničkog sučelja. Uspješno korisničko sučelje jest ono u kojem korisnik odmah uoči bitne stvari.



Slika 7-5. Dizajn korisničkog sučelja

Korisničko je sučelje dosljedno definiranoj tipografiji i paleti boja.

7.4.2. Dizajn navigacije

Dobar dizajn navigacije krucijalan je za *web site* jer bez navigacije stranica gubi sav smisao strukture i organizacije.

Jednostavnost je krucijalna za *website* navigaciju jer navigacija može biti prezentirana na više načina, od menija sa strane do menija u *headeru* ili *footeru* ili fiksnim pozicijama bilo gdje na stranici.

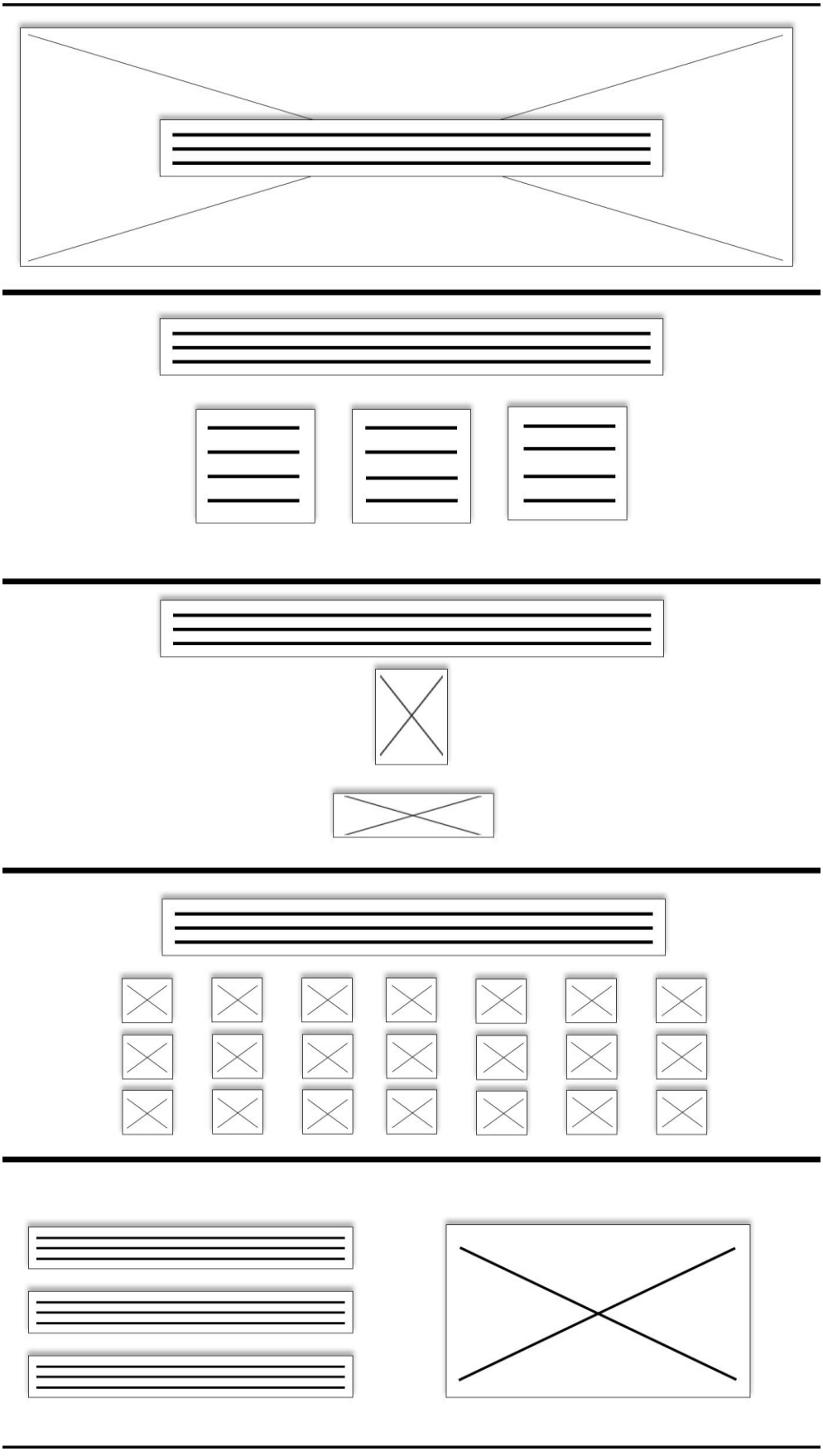
Za svoju sam navigaciju odabrao oblik fiksne navigacije koja se prikazuje kao mali *side* bar na desnoj strani *browsera*. Intuitivna je za moju ciljanu publiku te mislim da bi mi standardni oblik navigacije u *headeru* samo narušio dizajn i odvlačio pozornost od vizuala koje želim staviti u fokus na svakoj sekciji.

Korisnik se također navigira s pomoću *scrollanja* gore-dolje po stranici.

7.4.3. Dizajn informacija

Dizajn informacija tehnika je kako prezentirati informaciju da bi je korisnici što bolje i lakše razumjeli. Primjerice financijske podatke prezentiramo kroz grafove ili *pie charts*.

Dizajn informacija uključuje i grupiranje ili aranžiranje dijelova informacija kako bi bile lakše za uporabu. Listu npr. potrebnih podataka za registraciju grupirat ćemo u smislene cjeline – osobne informacije, adresa, dodatne informacije.



Slika 7-6. *Wireframe* stranice

7.5. Dizajn stranice

Sadržaj, funkcionalnost i estetika zajedno grade dobar dizajn stranice ispunjavajući ciljeve i zaključuju četiriju prethodnih planova.

Na mojoj stranici sadržaj i slike bit će raspoređeni tako da ostaje dosta bijelog prostora kako bi se ostavio dojam slobode i preglednosti te da stranica ne bude pretrpana jer bi je bilo teško čitati i doživjeti određene vizuale koji su ključni za neke akcije korisnika.

Stranica ni pod koju cijenu neće sadržavati oglase i reklame.

Sadržaj će biti sažet i konkretan, a korisnicima želimo pružiti maksimum informacija u što kraćem tekstu i vizualima.

Fontovi uporabljeni na *webu*:

- **Bangers Regular**

Slika 7-7. Primjer fonta uporabljenog na *webu*

Slika 7-8. Boja iz palete koja dominira na stranici

Slika 7-9. Finalni dizajn *web*-stranice

8. Prilike za unaprjeđenje

Aplikacija se može koristiti i daje tražene rezultate, ali to ne znači da nema prostora za unaprjeđenje.

Tri su stvari koje bi postavio kao potencijalne ciljeve unaprjeđenja korisničkog iskustva i funkcionalnosti:

- Mogučnost izvoza dizajna skate parka u 3D zapisu te mogučnost direktnog djeljenja na stranicu aplikacije
- Fizikalni proračun inercije i prosječne brzine skatera gdje bi se dizajn procjenio kroz funkcionalnu upotrebu
- Implementacija „Go Skate“ moda u kojem je moguče testirati skate park simulacijom vožnje po njemu.

8.1. Izvoz dizajna i prezentacija na stranici

Osim slike koja se može izvesti iz aplikacije koa je u rasterskom obliku u unaprjeđeno verziji postojala bi opcija izvoa cjelokupnog 3D dizajna u 3D obliku.

Taj dizajn bi se putem web stranice mogao poslati potencijalnom investitoru/klijentu. Korisnici koji prime link bi mogli pogledati taj skate park u 3D obliku što znaći da bi mogli u njega zoom-irati i rotirati ga.

Ta funkcionalnost bi prenosila jasniju poruku o dizajnu korisniku koji je prima.

Postojala bi opcija da dizajneri omoguče javni pregled posjetiteljima stranice i glasaju za njihov dizajn u svrhu promocijskih nagradnih igri ili pak ostave komentar za unaprjeđenje dizajna.

8.2. Funkcionalno unaprjeđenje kroz proračun inercije i brzine skatera

Kada bi se izradio algoritam koji bi računao inerciju i brzinu skatera koji kreče sa određenog elementa bilo bi lakše posložiti elemente da budu realnije raspoređeni.

Implementacijom tog algoritma bi se izbjegle pogreške u izgradnji i dizajnu koje bi štetile korisnicima skateparka.

Pogreške koje se mogu pojaviti su:

- Preveliki razmak između elemenata
- Premali razmak između elemenata

Prevelikim razmakom skater bi gubio brzinu što nije dobro jer bi se zatim morao odgurivat što rezultira većim trošenjem energije i ravnoteže.

Premalim razmakom skater nema vremena namjestiti položaj nogu na dasci za sljedeći trik što može rezultirati ozljedama.

Zaključak je taj da bi što preciznije trebalo računati inerciju i brzinu skatera kako bi se što bolje mogli rasporeditit elementi po skate parku kako nebi dolazilo do nepotrebnih ozljeda i prevelike potropšnje snage.

8.3. Implementacija simulacije vožnje kroz skate park

Kako bi se omogučilo testiranje dizajna na višoj razini implementirao bi mogučnost vožnje kroz skate park u ulozi skatera koji može raditit trikove na elementima.

Samim time bi se detaljnije uočile nepravilnosti i greške u dizajnu, te bi se došlo do bolje i savršenijeg rješenja dizajna skate parka.

Na ovaj zabavan način testiranja fokus bi padao na raspored elemenata i vozljivost skate parka.

Ime ovog moda bilo bi „Go Skate“.

Zaključak

U ovom je radu opisan jedan kompleksan projekt koji se sastoji od 3 velike faze kako bi se postigao rezultat.

Krenuo sam sa planiranjem modeliranja i teksturiranjem elemenata kako bih izgradio elemente kojima bih se koristio u aplikaciji. Nakon toga započeo sam s izradom same aplikacije koja je postala alat za dizajniranje *skate*-parkova. Na samome kraju napravio sam *web*-stranicu koja prezentira aplikaciju na *webu* i daje mogućnost korisnicima da je skinu, instaliraju i koriste se njome za vlastite potrebe.

Na kraju ovog projekta uočavam koliko je prostora ostalo za unaprjeđenje aplikacije, ali mislim da ona može postati profitabilna.

Na temelju dosadašnje verzije aplikacije mogu pokrenuti *crowfounding* kampanju u kojoj mogu prikupiti sredstva da nastavim s unaprjeđenjem aplikacije, unaprjeđenjem stranice i razradom marketinga.

Očekujem da u 2020. godini ovaj alat postane aktualan iz razloga što će *skateboarding* upravo te godine biti uvršten kao disciplina na Olimpijske igre. To će biti dovoljno dobar i pozitivan marketing za *skateboarding* kao kulturu i mislim da će nakon toga porasti potražnja za *skate*-parkovima nakon toga.

Ako se sve to odigra onako kako sam predvidio, ovaj bi alat definitivno mogao postati jedan od najzastupljenijih na tržištu.

Popis slika

Literatura

.

[1] Jesse James Garrett: The Elements of User Experience: User-Centered Design for the Web and Beyond, 2010.

[2] Priručnik - Standardi u primjeni internetske tehnologije, Visoka škola za primijenjeno računarstvo, Vedran Zdešić, Zagreb, 2009.

[3] PolyModeling With 3dsMax - Thinking Outside of the Box

Prilog

Završni rad može imati priloge, ali se oni ne prilažu uz pisanu verziju završnog rada, nego se mogu priložiti na završnom ispitu ako povjerenstvo na tom ispitu tako odluči. Važno je čuvati svu popratnu dokumentaciju koja je nastala pri izradi završnog rada.

S unutarnje strane na zadnjim koricama originala, kao i svake kopije završnog rada, pričvršćuje se CD s kompletnim završnim radom u izvornom formatu (npr. .doc) i .pdf formatu sa svom popratnom dokumentacijom i programima. Pri tome je obvezno da na tom CD-u postoji i dokument koji opisuje kako se rezultat njegova diplomskog rada (softver ili hardver) koristi (ili kako se npr. izvode mjerenja koja je opisao u radu). Ako je riječ o softveru, nužno je opisati i kako se programska podrška instalira.

ALGEBRA
VISOKO UČILIŠTE

NASLOV ZAVRŠNOG RADA

Pristupnik: Sven Bezi, 0275038907

Mentor: prof. dr. sc. Predrag Šuka