Parkkipaikkaa verkkoon visualisoiva järjestelmä

1. Johdanto

Tässä dokumentissa esitellään lyhyesti Karelia-amk:n Ketterä kehitysprojekti -kurssilla toteutetun parkkipaikkaa verkkoon visualisoivan järjestelmän toimintaa. Tavoitteena on että dokumentin pohjalta kehitystyötä on mahdollista tulevaisuudessa jatkaa, mikäli esimerkiksi kampuksen parkkia saadaan kuvaamaan kiinteä kamera. Tässä työssä auttavat myös tuotetusta koodista löytyvät kommentit.

Projektiryhmään kuuluivat: Sami Kujala, Sami Karhunen, Joni Rokkanen, Tuomas Keinänen, Jussi Piirainen sekä Jani Rautalin.

2. Scene

Tässä luvussa käsitellään projektin scenen ulkoasua, objekteja sekä vuoden- ja vuorokaudenajan esittämiseen liittyviä asioita.

2.1. Parkkialue ja objektit

Sceneen on mallinnettu yksinkertaistettu ja pienennetty versio Karelia-amk:n Wärtsilä -kampuksen opiskelijoiden parkkialueesta. Mallinnuksessa on käytetty itse tehtyjä peliobjekteja (esim. koulurakennus, terrainit, parkkikyltit), Unityn perusassetteja sekä Unityn Asset Storen ilmaisia assetteja (esim. autojen runko, katuvalot), puut. Visuaalinen ilme pyrittiin luomaan sarjakuvamaiseksi (kuva 1).

Kuva 1, skenen visuaalinen ilme.

2.2. Vuoden- ja vuorokaudenajat

Vuorokauden ja vuodenajat vaihtuvat getTime-skriptin mukaan. Järjestelmä hakee käyttäjän tietokoneen aikatiedot ja muuttaa scenestä sen perusteella vuorokauden- sekä vuodenajan. Vuoden- ja vuorokauden aikoja voi halutessaan muokata myös editorista laittamalla overriden päälle ja muokkaamalla arvoja manuaalisesti (tulee asettaa ennen käynnistämistä), molempien arvot ovat välillä 1-4, vastaten kevät, kesä, syksy, talvi sekä aamu, päivä, ilta/iltapäivä, yö.

Käytännössä vuodenaikojen muutokset vaihtavat käytettävän terrainin ja puuobjektit vastaamaan kyseistä vuodenaikaa. Talvella taivaalle ilmestyy myös pilviä ja lumisadetta. Vuorokaudenaikojen raja-arvot vaihtelevat vuodenajan mukaan (talvella lyhempi päivä). Vuorokaudenajan mukaan määrittyvät puolestaan auringon sijainti sekä scenen valaistus.

GetTime-skriptistä löytyvä “demo-mode” mahdollistaa auringon liikkeen nopeuttamisen “demo cycle speed” muuttujan määrittämällä nopeudella.

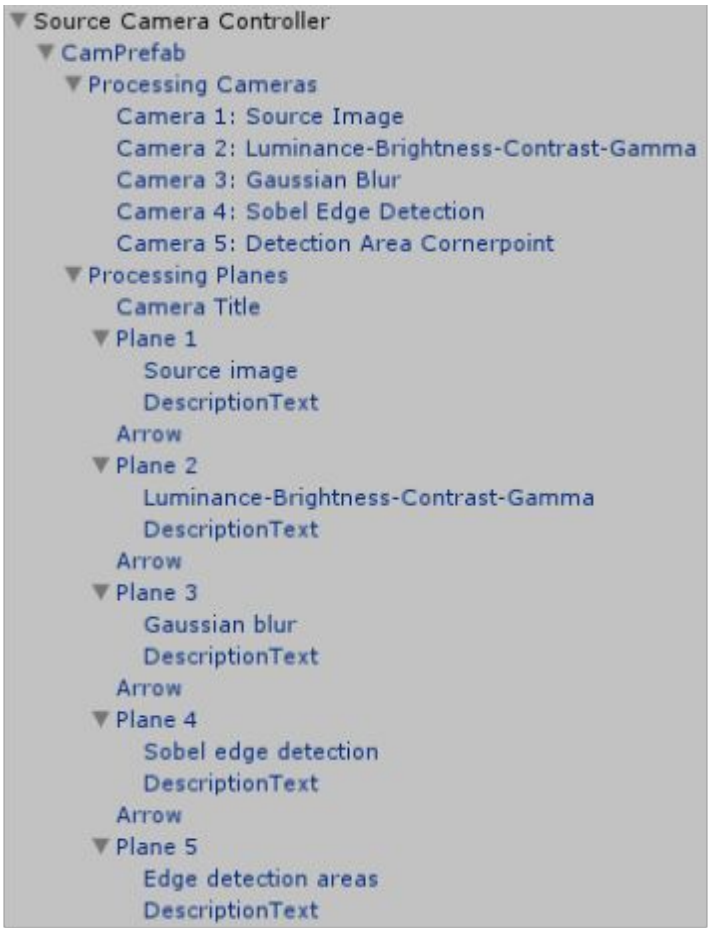
3. Reaaliaikainen visualisointi

Tässä luvussa käsitellään projektin ydintoiminnallisuutta: kuinka valokuvan avulla saadaan autot liikkumaan.

3.1 Reunantunnistus

3.1.1 Toimintakuvaus

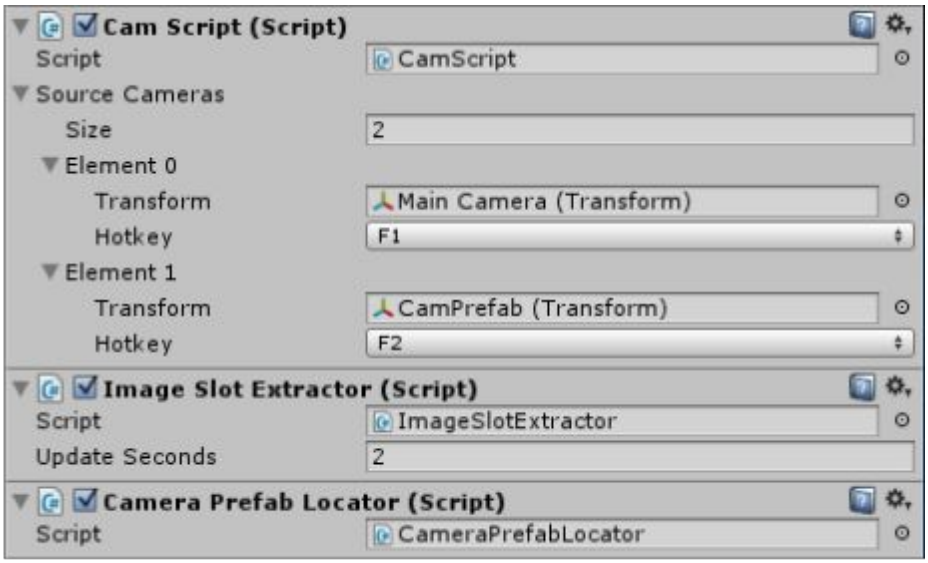
Reunantunnistuksen perusyksikkönä parkkipaikkaskenessä toimii Assets-pääkansiosta löytyvä lähdekameraa simuloiva objektisäiliö CamPrefab (kuva 2). Aliluvuissa säiliön toiminta selitetään aliobjekteittain ja kuhunkin objektiin kytkettyjen skriptien toiminta kuvataan yleistasolla. Skriptit löytyvät projektin Assets/Detection-hakemistosta, ellei kuvauksessa mainita toisin.



Kuva 2, CamPrefab-lähdekamerasäiliön hierarkia.

Taso 1: Source Camera Controller

Skeneen lisättävä uusi kamerasäiliö tulee sijoittaa skenehierarkian ylimmällä tasolla olevan Source Camera Controller -objektin (kuva 3, jatkossa SCC) ensimmäisen sukupolven lapsiobjektiksi. Skenen kamerasäiliöissä tapahtuvan lukumäärän tai SCC:n alaisen keskinäisen järjestyksen muututtua SCC tulee kertaalleen valita Unityn hierarkiaikkunassa. Tällöin SCC:hen kytketyn CamScriptin toiminnallisuutta Assets/Editor-kansiossa laajentava skripti SourceCamSelector_EditorExtension päivittää uuden säiliökonfiguraation automaattisesti osaksi reunantunnistustoimintoa.



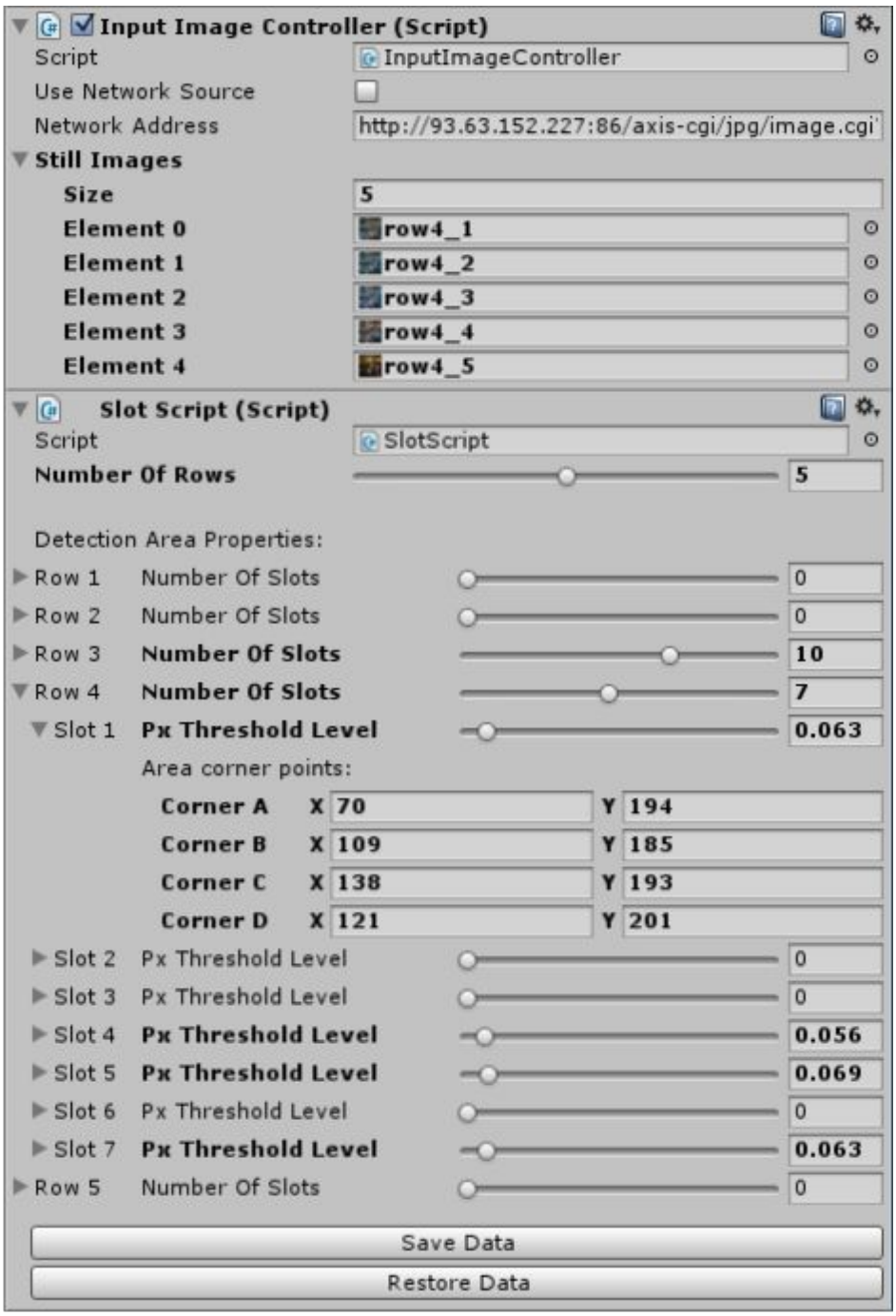
Kuva 3, Source Camera Controller.

Skriptit:

- CamScript. Skripti ylläpitää pääkameran ja lähdekamerasäiliöiden lukumäärää, kytkee näihin pikanäppäimet, joilla kukin kamera on käyttäjän aktivoitavissa sekä toteuttaa pikanäppäintoiminnallisuuden. Pääkamera kuvaa skenen parkkipaikkamallinnosnäkymää, lähdekamerasäiliöt taas kukin tuottamansa lähdekuvan kuvaprosessoinnin eri vaiheita. Oletuspikanäppäimet juoksevat ylöspäin alkaen funktionäppäimestä F1.
- ImageSlotExtractor. Skripti ylläpitää keskitetysti eri kamerasäiliöiden lähdekuviin liittyviä parkkiruutumääritystietoja. Näiden tietojen jakamisen autojen liikeskriptille ohessa skripti määrittää ja tulostaa konsoliin kustakin kamerasäiliön lähdekuvasta prosessoidusta binäärireunantunnistuskuvasta tunnistustiedot sekä välittää myös nämä tiedot autojen liikeskriptille käyttäjän syöttämin aikavälein.
- CameraPrefabLocator. Skripti valvoo ajastetusti skeneen lisättyjen lähdekamerasäiliöiden oikeaa sijaintia SCC:n alla tulostaen tarvittaessa konsoliin virheilmoituksen ja korjausohjeen.

Taso 2: CamPrefab

Kamerasäiliöhierarkian ylimpänä on isäntäobjekti (kuva 4, kuvassa 2 oletusnimellä “CamPrefab”), jonka nimi on vapaasti muokattavissa. Nimi toimii samalla lähdekameran yksilöivänä nimenä, jonka perusteella kamerasäiliön lähdekuvaprosessoinnin visuaalinen debuggaustoiminto on skenessä erotettavissa muista eri nimisistä kamerasäiliöistä. Lisäksi nimi määrittää käyttäjän syöttämien reunantunnistusalueiden tiedostoon tallennus- ja lataustoimintojen kohdetiedoston nimen.



Kuva 4, CamPrefab.

Skriptit:

- InputImageController. Skripti määrittelee kuvalähteen.

  Verkkokuvalähde:
  Skripti noutaa IP-kameran verkkoon päivittämiä kuvia pyrkien lataamaan niitä kuvan reunantunnistustoiminnon käyttöön Source Camera Controller -objektiin kytketyn ImageSlotExtractor-skriptin määrittelemällä päivitysaikavälillä.

  Etäverkkopalvelimen lähdekuvaa käytettäessä palvelimen tulee olla konfiguroitu sallimaan domainin ulkopuoliset resurssipyynnöt (Cross-Origin Resource Sharing, CORS), mikäli verkkokuvalähdettä aiotaan käyttää WebGL-buildissa. Mikäli WebGL-version käyttämä verkkokuvalähde sijaitsee samassa domainissa tai projektia ajetaan Unityn editoritilassa kuvahaku toimii myös ilman CORS-toteutusta.

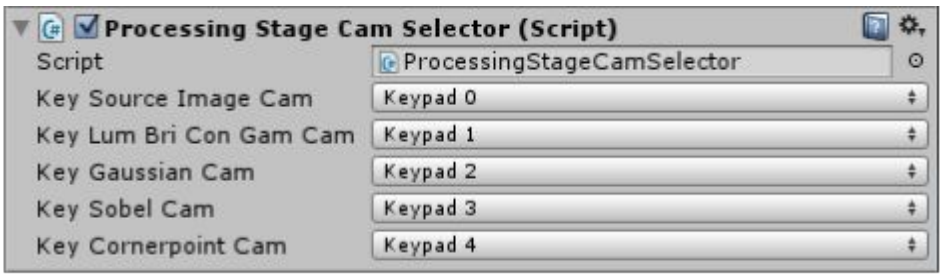
  Still-kuvalähde:
  Demokäytössä kuvia on mahdollista syöttää vapaavalintainen määrä still-kuvatiedostoina paikallisesta tallennuskohteesta. Vaihtelu kuvan reunantunnistuksessa käytettävien still-kuvien välillä tapahtuu skriptissä kiinteästi määrittelyillä näppäimillä alkaen numeronäppäimestä 1 eteenpäin. Numeronäppäin 1 valitsee still-kuvataulukon ensimmäisen solun kuvan, numeronäppäin 2 toisen solun kuvan ja niin edelleen. Kuvavalinta numeronäppäimillä kohdistuu samanaikaisesti kaikkiin still-kuvia käyttäviin skenen kamerasäiliöihin, mikäli säiliössä on määritelty valittua näppäinindeksiä vastaava kuvataulukon solu. Mikäli still-kuvia on määritelty säiliössä enemmän kuin yhdeksän, aktivointipainikkeet jatkuvat numeropainikkeiden 1-9 jälkeen sivun http://docs.unity3d.com/ScriptReference/KeyCode.html ilmaisemin painikkein alkaen kohdan Alpha9 jälkeen.
- SlotScript. Skripti esittelee kehittäjän määrittelemän SlotData-tyypin muuttujan, jonka sisällön Assets/Editor-kansiossa sijaitseva, toiminnallisuutta laajentava SlotPropertyDrawer-skripti avaa Unityn inspektoriin. Skripti määrittää lähdekuvan parkkiruutujen tunnistuskynnystasot sekä -alueet rivi- ja paikkanumeroin. Skripti SlotDataFileAccessor_EditorExtension laajentaa edelleen toiminnallisuutta lisäämällä tallennus- ja palautuspainikkeet, joilla kertaalleen määritellyt tunnistusasetukset voi viedä tiedostoon ja tuoda tiedostosta. Tiedostosijainti on Assets/SlotData-alikansiossa, jonne asetustiedosto luodaan kamerasäiliön mukaan nimettynä tekstitiedostona.

Taso 3.1: Processing Cameras

Objekti (kuva 5) sisältää kameran kunkin lähdekuvankäsittelyvaiheen esittämiseksi käyttäjälle. Kameroita on yhteensä viisi kappaletta:

1. Raakalähdekuvakamera
2. Harmaasävy-, kirkkaus-, kontrasti- ja gamma-prosessointivaihekamera
3. Gauss-sumennusvaihekamera
4. Sobel-reunantunnistusvaihekamera
5. Tunnistusalueiden kulmapistepiirtokamera.

Kukin kamera on kohdistettu skenessä omaan taso-objektiinsa, johon lähdekuvan prosessointivaihe piirtyy kameran esitettäväksi käyttäjälle.



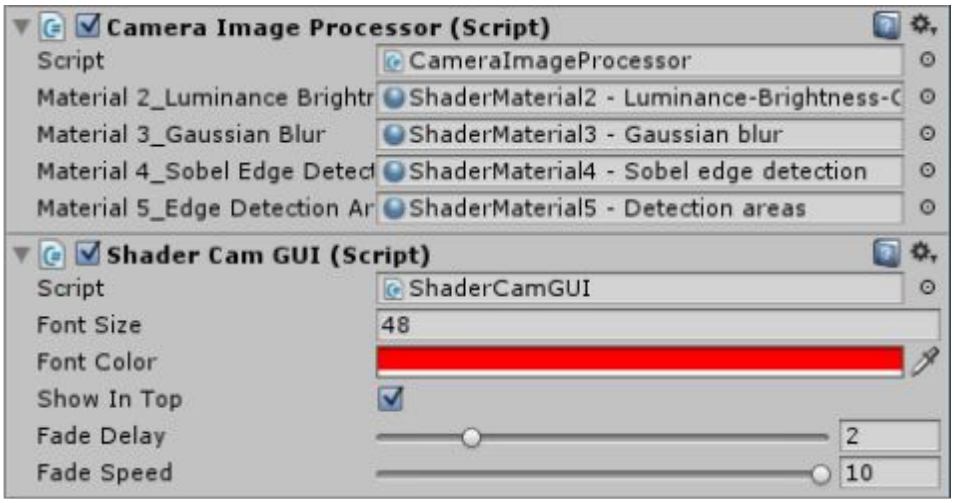
Kuva 5, Processing Cameras.

Skriptit:

- ProcessingStageCamSelector. Skripti määrittää ja toteuttaa pikanäppäimet lähdekuvakamerasäiliön prosessointivaihekameroiden aktivoimiseksi, kun kyseinen kamerasäiliö on ensin valittu Source Camera Controller -objektin CamScriptiin määritetyllä pikanäppäimellä. Oletuspainikkeina ovat keypad 0 - keypad 4.

Taso 3.1.1: Camera 1: Source Image

Objekti (kuva 6) on kamerasäiliön raakaa lähdekuvaa esittävään tasoon kohdistettu kamera, joka välittää kuvatiedon eteenpäin muille kuvaprosessointivaihetasoille käyttäjän määrittämin prosessointiasetuksin.



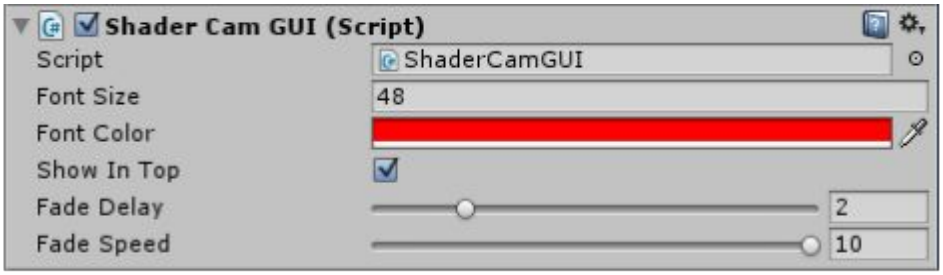
Kuva 6, Camera 1: Source Image.

Skriptit:

- CameraImageProcessor. Skripti suorittaa lähdekuvan välittämisen prosessointivaihe vaiheelta OpenGL-shaderohjelmien läpi kuvaprosessointivaihetasoille.
- ShaderCamGUI. Skripti piirtää kyseinen kuvaprosessointivaihekamera aktivoitaessa kameran ruudulle välittämän kuvan päälle häivytettävän infotekstin, joka kertoo kulloisenkin valitun kamerasäiliön ja kuvaprosessointivaiheen nimet käyttäjän määrittämin tekstiasetuksin.

Taso 3.1.2: Camera 2: Luminance-Brightness-Contrast-Gamma

Objekti (kuva 7) on kamerasäiliön raa'asta lähdekuvasta harmaasävy-, kirkkaus-, kontrasti- ja gamma-prosessoidusta läpi kulkeneen kuvan esittämään kuvaprosessointivaihetasoon kohdistettu kamera.



Kuva 7, Camera 2: Luminance-Brightness-Contrast-Gamma.

Skriptit:

- ShaderCamGUI. Katso tason 3.1.1 ShaderCamGUI-kuvaus.

Taso 3.1.3: Camera 3: Gaussian Blur

Objekti (kuva 7) on kamerasäiliön harmaasävy-, kirkkaus-, kontrasti- ja gamma-prosessoidusta kuvasta Gauss-sumennusvaiheen läpi kulkeneen kuvan esittämään kuvaprosessointivaihetasoon kohdistettu kamera.

Skriptit:

- ShaderCamGUI. Katso tason 3.1.1 ShaderCamGUI-kuvaus.

Taso 3.1.4: Camera 4: Sobel Edge Detection

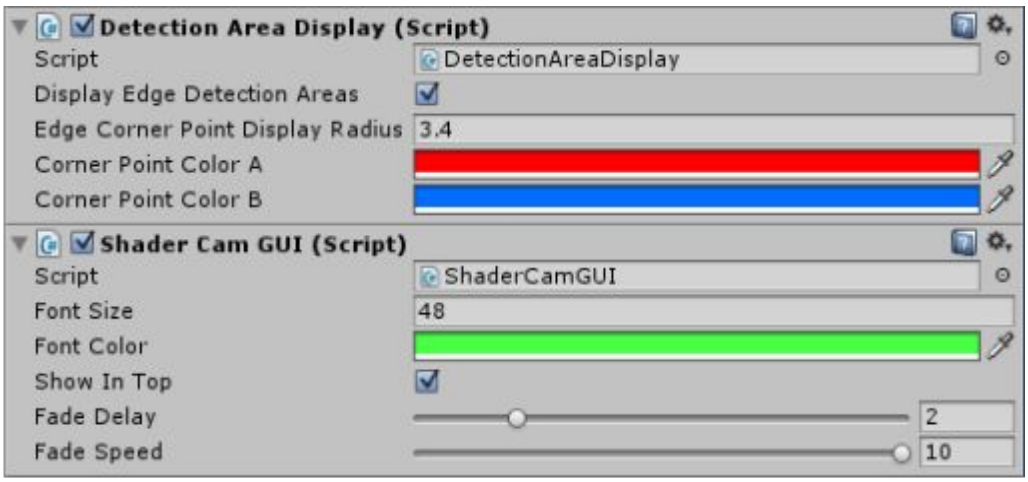
Objekti (kuva 7) on kamerasäiliön Gauss-sumennusprosessoidusta kuvasta Sobel-reunantunnistusvaiheen läpi kulkeneen kuvan esittämään kuvaprosessointivaihetasoon kohdistettu kamera.

Skriptit:

- ShaderCamGUI. Katso tason 3.1.1 ShaderCamGUI-kuvaus.

Taso 3.1.5: Camera 5: Detection Area Cornerpoint

Objekti (kuva 8) on kamerasäiliön Sobel-reunantunnistusprosessoidusta kuvasta tunnistusalueiden kulmapistepiirtovaiheen läpi kulkeneen kuvan esittämään kuvaprosessointivaihetasoon kohdistettu kamera.



Kuva 8, Camera 5: Detection Area Cornerpoint.

Skriptit:

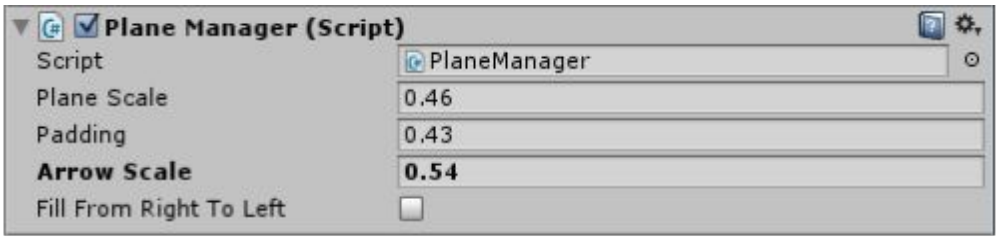
- DetectionAreaDisplay. Skripti kokoaa käyttäjän CamPrefab-olion SlotScriptiin määrittämiä kulmapisteitä vastaavan väritekstuurin ja lähettää sen kulmapistepiirto-shaderohjelmalle. Skripti määrittää myös muita käyttäjän muokattavissa olevia kulmapistepiirtoasetuksia, joiden perusteella kulmapistekuvaprosessointivaihetasolla esitetään kuvan kulmapisteet.

Taso 3.2: Processing Planes

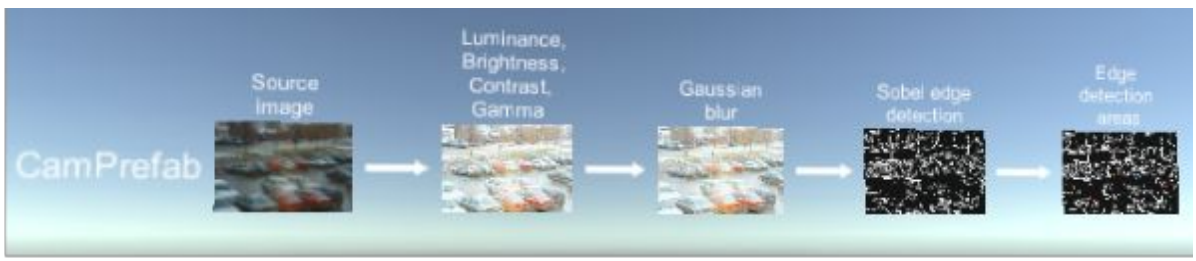
Objekti (kuva 9) toimii isäntänä visuaalista palautetta kuvaprosessoinnista skenessä tuottaville objekteille, joihin kuuluu kamerasäiliön nimitekstiobjekti, kuvaprosessointivaihetasot kuvaustekstiobjekteineen sekä kunkin vierekkäisen kuvaprosessointivaihetasoparin välillä oleva suuntanuoli (kuva 10). Kuvaprosessointivaihetasoja on Processing Cameras -objektin kameroiden tapaan yhteensä viisi kappaletta:

1. Raakalähdekuvataso
2. Harmaasävy-, kirkkaus-, kontrasti- ja gamma-prosessointivaihetaso
3. Gauss-sumennusvaihetaso
4. Sobel-reunantunnistusvaihetaso
5. Tunnistusalueiden kulmapistepiirtotaso.

Tasoille piirtyvät yllä olevan listauksen mukaiset kuvaprosessointivaiheiden tulokset.



Kuva 9, Processing Planes.



Kuva 10, kuvaprosessointivaiheiden visuaalinen näyttö.

Skriptit:

- PlaneManager. Skripti toteuttaa kuvaprosessointivaihetasojen ja näihin liittyvien visuaalisten otsikkoteksti- ja suuntanuoliapuobjektien kokoskaalauksen ja keskinäisen sijoittelun käyttäjän antamien arvojen mukaisesti.

Taso 3.2.1: Camera Title

Objekti (kuva 11) esittää kamerasäiliön nimen skenessä osana kuvaprosessointiketjun visualisointia.

Kuva 11, Camera Title.

Skriptit:

- PlaneTitleManager. Skripti siirtää kamerasäiliön nimen tekstiobjektin tekstisisällöksi.

Tasot 3.2.2, 3.2.4, 3.2.6, 3.2.8, 3.2.10: Plane N

Objekti toimii isäntänä kuvaprosessointitasolle sekä tason kuvaustekstiobjektille.

Taso 3.2.2.1: Source Image

Kuvaprosessointitaso-objekti (kuva 12) näyttää kamerasäiliön raakalähdekuvan.

Kuva 12, Source Image.

Skriptit:

- PlaneCameraBinder. Skripti asettaa tasoa vastaavan prosessointivaihekameran sijainnin ja konfiguroi kuvakoko- ja -suhdeasetukset niin, että taso täyttää kameran näkymän. Lisäksi skripti asettaa kamerasäiliön nimen sekä tason kuvaustekstin prosessointivaihenäkymän päällä näytettäväksi häivytystekstiksi.
- PlaneTextBinder. Skripti kohdistaa kuvaprosessointitason kuvaustekstiobjektin tason yläpuolelle sivusuuntaiseen keskikohtaan sekä skaalaa tekstikoon tason kokoon suhteutettuna.

Tasot 3.2.2.2, 3.2.4.2, 3.2.6.2, 3.2.8.2, 3.2.10.2: DescriptionText

Tekstiobjekti on visuaalinen esitys sisarusobjektina olevan kuvaprosessointitason kuvaustekstistä.

Tasot 3.2.3, 3.2.5, 3.2.7, 3.2.9: Arrow

Kuvaobjekti on visuaalinen esitys vierekkäisten kuvaprosessointitasojen välisestä suuntanuolesta.

Tasot 3.2.4.1, 3.2.6.1, 3.2.8.1, 3.2.10.1: <Prosessointivaihenimi>

Kuvaprosessointitaso-objekti (kuva 13) näyttää kamerasäiliön prosessointivaiheitta vastaavan prosessoidun kuvan. Prosessointivaiheet ovat:

- harmaasävy-, kirkkaus-, kontrasti- ja gammaprosessointivaihe
- Gauss-sumennusvaihe
- Sobel-reunantunnistusvaihe
- Tunnistusalueiden kulmapistepiirtovaihe.

Kuva 13, Luminance-Brightness-Contrast-Gamma

Skriptit:

- PlaneCameraBinder. Katso tason 3.2.2.1 kuvaus.
- PlaneTextManager. Katso tason 3.2.2.1 kuvaus
- SetUnlitShader. Skripti korvaa ajonaikaisesti luodun tasomateriaalin oletusshaderin sopivan kuvavalaistuksen tuottavalla shaderilla.

3.1.2 Toteuttaminen uudella kuvalähteellä

Lähtötilanteessa skenessä ei ole kamerasäiliöitä ja Source Camera Controller -objekti (jatkossa SCC) on tyhjä.

1. Aseta CamPrefab skeneen ja siirrä se SCC:n alle lapsiobjektiksi. Nimeä prefab halutessasi lähdekameraa kuvaavalla nimellä.
2. Valitse Unityn hierarkiaikkunasta SCC. Kamerasäiliö päivittyy inspektorinäkymään CamScriptin alle.
3. Valitse hierarkiaikkunasta tuomasi prefab ja aseta inspektorissa still-kuvataulukon koko ja kuvatekstuurit tai valitse verkkokuvalähde ja syötä verkko-osoite.
4. Aseta inspektorissa SlotScriptin alle haluamasi määrä tunnistusrivejä.

   Aseta riveittäin haluamasi määrä tunnistuspaikkoja. Aseta kullekin tunnistuspaikalle reunantunnistuspikselikynnystaso, joka määrittää prosessoidun binäärikuvan tunnistuskertoimen. Arvovälille [0...1] interpoloitu kerroin määrittää, kuinka paljon reunapikseleitä kyseiseltä tunnistusalueelta tulee vähintään löytyä suhteessa alueen kokonaispikselimäärään, että tulos tulkittaisiin varatuksi parkkiruuduksi. Esimerkiksi kerroin 0.2 siis vastaa 20 % tunnistusalueen kokonaispikselimäärästä.

   Aseta kullekin tunnistuspaikalle lähdekuvasta nelikulmion rajaaman tunnistusalueen kulmapistekoordinaatit. Koordinaatti (0, 0) sijaitsee kuvan vasemmassa yläkulmassa. Neljä samaa (X, Y)-koordinaattipistettä syöttämällä on mahdollista ohittaa kyseisen tunnistuspaikan tarkastelu.

   Tunnistuspaikkadata on mahdollista tallentaa tiedostoon Save Data -painikkeella ja ladata tiedostosta Restore Data -painikkeella.
5. Käynnistä skene. Mikäli käytössä on verkkokuvalähde, kuvien tulisi alkaa päivittyä automaattisesti. Jos käytössä on paikallinen kuvatiedostosarja, numeropainikkeet alkaen painikkeesta 1 vaihtavat kuvien välillä.

   Unityn konsoliin tulostuu tietoa kuvantunnistuksen tuloksista. Kuvaprosessointitasot antavat myös visuaalista palautetta kuvaprosessointivaiheista. Kuvaprosessointia tekevien shaderohjelmien parametrit ovat säädettävissä valitsemalla jokin ShaderMaterial2 - ShaderMaterial5 -materiaaleista projektitiedostonäkymäikkunassa kansiosta Assets/Detection/Materials ja säätämällä arvoja inspektorissa. Arvot ovat kaikille kuvalähteille globaalisti yhteiset.

   Oletuspainikkeella F2 pääset siirtymään lisäämäsi kameran kuvaprosessointivaiheiden tarkasteluun. Kyseisessä tilassa näet eri prosessointivaiheet oletuspainikkeilla keypad 0 - keypad 4. Erityisesti viimeinen, kuvatunnistusalueiden kulmapisteet binääriprosessoidun kuvan päälle piirtävä prosessointivaihe on hyödyllinen kuvantunnistusalueiden määritysten tarkastuksessa. Oletuspainike F1 palauttaa takaisin parkkipaikkamallinnosnäkymään.

3.2 Autojen liike

Autojen liikkuminen ruutuihin tapahtuu driveLineController.cs scriptin kautta. Liikkumisessa hyödynnetään Unityn NavMesh-työkaluja. Varatut parkkiruudut kommunikoidaan reunantunnistus-skriptistä tähän skriptiin, joka säätelee rows-muuttujan avulla boolean arvoja, joka määrittää onko ruudussa auto vai ei.. Tätä tarkkailee checkSpots() -metodi. Autojen liikettä rajoittavat NavMeshObstaclet. Jokaisella parkkiruudulla on NavMeshObstacle, joka on päällä (estää liikkumisen alueella), jos ruudussa on, tai tulee olemaan (auto on liikkeessä), auto. Lisäksi driveLineController asettaa ensimmäisessä päivityksessä jo ruuduissa olevat autot.

Paikan ollessa varattu, kytketään paikan NavMeshObstacle pois päältä, mahdollistaen auton siirtymisen ruutuun. Tämän jälkeen auto spawnataan carSpawniin ja auton NavMeshAgentille annetaan kohteeksi parkkiruutu. Parkkiruudut ovat numeroituja ja ruutuun spawnataan ruudun numerolla varustettu auto.

Kun ruutu vapautuu, peruutetaan auto pois ruudusta ja annetaan NavMeshAgentille kohteeksi carSpawn ja asetetaan auton tagiksi “kill”. Osuessa carSpawnin collideriin “kill” -tagilla varustetut peliobjektit tuhotaan.

NavMesh-navigointia on avustettu skriptissä mm. autojen rotaation säätelemisellä.

4. Käyttöliittymä

Käyttöliittymä on yksinkertaisuudessaan muutamia rivejä tekstejä ja toggle -painike ohjeiden näyttämiselle ja piilottamiselle. UI Canvasta voidaan tutkia scene ruudulla, josta selviää että tuo on valtava alue, joka skaalautuu ruudun koon mukaan ja on aika etualalla sovelluksen sisällä liikuttaessa. Fontit yms. on erittäin helppo muuttaa valitsemalla elementti hierarkiapuusta ja muuttamalla sen asetuksia.