LAPPEENRANNAN TEKNILLINEN YLIOPISTO
LUT School of Energy Systems
LUT Kone

PLC-OHJELMOINNIN OPETUKSEN PELILLISTÄMINEN

GAMIFYING THE EDUCATION OF PLC-PROGRAMMING

Lappeenrannassa 20.11.2018
Perttu Juvonen
Tarkastajat TkT Hamid Roozbahani, TkT Heikki Handroos
Ohjaajat TkT Lauri Luostarinen, TkT Hamid Roozbahani

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
LUT Energiajärjestelmät
LUT Kone

Perttu Juvonen

PLC-ohjelmoinnin opetuksen pelillistäminen

Kandidaatintyö

2018

34 sivua, 17 kuvaa ja 21 liitettä

Tarkastajat: TkT Hamid Roozbahani, TkT Heikki Handroos

Ohjaajat: TkT Lauri Luostarinen, TkT Hamid Roozbahani

Hakusanat: pelit, pelillistäminen, plc, ohjelmointi, opetus, unity 3D, iec, 61131-3, codesys

Tässä kandidaatintyössä tavoitteena on ollut luoda pelillinen elementti Lappeenrannan teknillisen yliopiston mekatroniikan kurssin harjoitustyön avuksi. Kurssin harjoitustyön tavoitteena on luoda katsaus PLC-ohjelmointiin IEC61131-3 mukaisella ohjelmointikielellä ja sen toteuttamiseen käyttäen Codesys-ohjelmistoympäristöä. Työn alussa on tiivis katsaus peleihin ja pelillistämiseen käsitteenä ja näiden hyödyntämiseen opetuskäytössä. Hankittujen tietojen pohjalta on toteutettu pelillinen elementti käyttäen Unity-3D pelinkehitysympäristöä. Ympäristö on valittu vertaillen muita käyttötarkoitukseen soveltuvia ohjelmistovaihtoehtoja.

Toteutettavan ohjelmiston periaatteet, sisältö ja toteutus ovat esitelty työssä. Ohjelma on toteutettu käyttäen hyvän ohjelmoinnin periaatteita ja ohjelmiston uudelleenkäytettävyys ja jatkokehityskelpoisuus päätavoitteina. Työn lopussa kuvataan tuotetun ohjelmiston toimintaperiaatteet ja käyttäytyminen. Tuotetun ohjelman kehittämiselle ja laajentamiselle jatkossa esitetään vaihtoehtoja ja tavoitteita.

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
LUT Mechanical Engineering

Perttu Juvonen

Gamifying the education of PLC-programming

Bachelor's thesis

2018

34 pages, 17 figures and 21 appendixes

Examiners: D. Sc. (Tech) Hamid Roozbahani, D. Sc. (Tech) Heikki Handroos

Supervisors: D. Sc. (Tech) Lauri Luostarinen, D.Sc. (Tech) Hamid Roozbahani

Keywords: gamification, plc, programming, teaching, unity, iec, 61131-3, codesys

In this bachelors thesis, the aim has been to create a gamified element for an exercise assignment on a mechatronics course at Lappeenranta University of Technology. The aim of the course's assignment is to provide a look at PLC programming with the programming languages specified by IEC61131-3 and its implementation using the Codesys development environment. At the beginning of the thesis there is a brief look into games and gamification concepts as well as to the utilization of them in the teaching environment. Based on the research on these subjects, a gamified element was implemented using the Unity-3D game development software. The environment has been selected after comparing it and other selected software alternatives available.

The principles, content and implementation of the software are presented in this thesis. The program is implemented following the principles of good programming, good software re-usability and further development abilities as the main priorities. At the end of the thesis, the functionality and behavior of the produced software are described. Some O
options and suggestions for developing and expanding the program further in the future are presented.

SISÄLLYSLUETTELO

LIITTEET

SYMBOLI- JA LYHENNELUETTELO

AI – Artificial Intelligence, Keinoäly

C# – Microsoftin kehittämä objektiorientoitunut ohjelmointikieli

GVL – Global Variable List, globaali muuttujalista

GPL – General Public License

IO – Input-Output, laitteen/ohjelman sisääntulot ja ulosmenot

NPC – Non-Playable Character, ei pelattava hahmo

NVL– Network Variable List, nettimuuttujalista

POU – Program Organizational Unit, ohjelmien rakenneyksikkö

PLC – Programmable Logic Controller, ohjelmoitava logiikkaohjain

UDP – User Datagram Protocol, yhteydetön verkkoprotokolla

1 JOHDANTO

Pelit ja pelillisyys ovat nykypäivänä integraalinen osa jokaisen elämää. Lapset oppivat elämän varrella hyödyllisiä taitoja osana leikkiä ja pelejä esimerkiksi kotia leikkiessä opitaan kotitalouden hoitoon liittyviä toimia aivan huomaamatta. Tietokoneiden ja mobiililaitteiden yleistyessä informaation saatavuus ja muodot ovat muuttuneet huomattavasti. Päivittäiset asiat kuten mainonta ja ostokset, urheilu ja terveys kuten myös opetuksen menetelmät ovat siirtyneet digitaaliseen maailmaan (Dombrowski & Lochner 2015).

Tässä kandidaatintyössä käsitellään pelillistämistä, sen keinoja ja niiden hyödyntämistä Lappeenrannan teknillisen yliopiston mekatroniikan kurssin PLC-ohjelmoinnin harjoitustyön yhteydessä. Kurssin harjoitustyön toteutus voisi olla nykyistä innostavampi ja havainnollisempi, minkä saavuttamista voidaan tavoitella esimerkiksi pelillistämiselementtejä hyödyntämällä.

Oppiminen ja uusien aiheiden käsittely saadaan tehostumaan pelillisiä tai leikillisiä menetelmiä käyttäen, mistä syystä pelillistäminen on yleistyvä ilmiö nykypäivän koulutustilanteissa ja arkielämässä. Leikki motivoi ja innostaa tekemään toimintoja, joita tehdessä opitaan erilaisia taitoja ja menetelmiä. Pääpaino siirtyy pelillistäessä oppimisesta aiheen käsittelyyn positiivisessa ilmapiirissä.

1.1 Työn tutkimustavoite ja -kysymykset

Työn päätavoitteena on kehittää toimiva pelillinen elementti Lappeenrannan teknillisen yliopiston mekatroniikan kurssin harjoitustyön avuksi. Harjoitustyössä opiskelijaryhmien tavoitteena on toteuttaa ohjausjärjestelmä parkkihalliin käyttäen PLC ohjelmistoa. Harjoitustyö pohjautuu nykyisessä muodossaan pelkästään Codesys-ympäristön tarjoamiin yksinkertaisiin elementteihin, minkä takia pelillistämisellä voidaan saavuttaa paremmin toimiva opetuskokonaisuus. Ohjelmiston toteuttamisen ohessa tutkitaan pelillistämisen keinoja opetuksen tehostamisessa ja hyödynnetään näitä toteutetussa ohjelmistossa. Tarkoituksena työssä on vastata siihen, kuinka opetuksen pelillistäminen voidaan toteuttaa ja mitä etuja pelillistämisellä haetaan.

1.2 Työn rajaaminen ja rakenne

Työssä keskitytään pääasiassa sovelluksen tuottamiseen ja pelillistämisen peruskäsitteen avaamiseen. Tuotettu sovellus on tarkoitus ottaa opetuskäyttöön mekatroniikan kurssilla sen valmistuttua. Työn pääpaino ei ole vaihtoehtoisten ratkaisujen etsimisessä pelillistämisen toteuttamiseen, mutta niitä sivutaan ohjelmiston toteuttamisen ohessa.

Sovelluksen valmistuttua ei tässä työssä tutkita sovelluksen hyödyntämisen vaikutusta kurssin opetukseen, vaan tämä jätetään mahdollisen tulevan tutkimuksen aiheeksi.

Työ rakentuu pelillistämisen peruskäsitteen selostamiseen, aihepiirin avaamiseen, ohjelmistoympäristön valintaperusteiden ja vaihtoehtojen esittelyyn. Näiden pohjalta siirrytään ohjelmiston toteutukseen, toiminnallisuuden esittelyyn ja lopuksi saavutettujen tulosten kertaamiseen kokonaisuutena.

2 PELIT JA PELILLISTÄMINEN

Tässä kappaleessa avataan pelejä ja pelillistämistä käsitteinä ja sidotaan niitä tehtävään työhön. Työ lähdetään rakentamaan pelillistämisen periaatteiden ympärille.

2.1 Miksi pelit kiinnostavat

Mikä tekee leikeistä ja peleistä viihdyttäviä ja kiinnostavia? Mikä saa ihmiset käyttämään huomattavia aikoja pelaamisen parissa?

Hunicke, LeBlac ja Zubek (2004) esittelevät tutkimuksessaan määritelmiä sille, mitkä tekevät peleistä ”hauskoja”. Heidän esittämänsä luokitukset ovat listattuna:

1. Tuntemus
2. Fantasia
3. Narratiivi
4. Haaste
5. Kumppanuus
6. Löytäminen
7. Ilmaiseminen
8. Nöyrtyminen

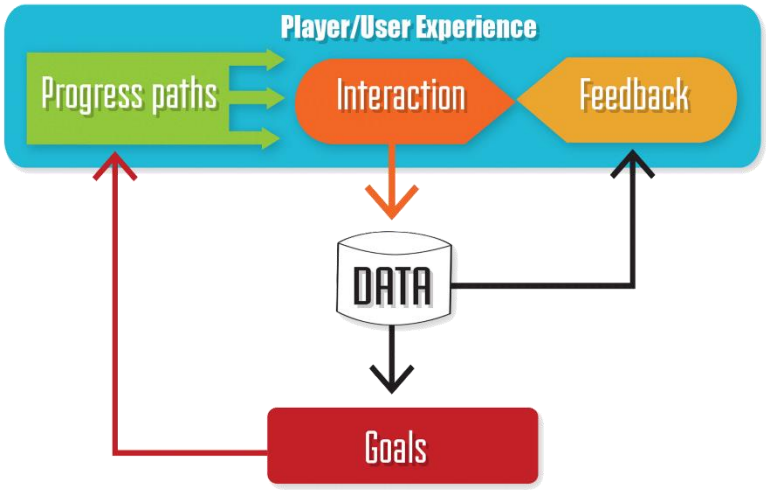
Heidän mukaansa kiinnostava ja hauska peli tai pelillistetty toiminta perustuu yhdelle tai useammalle näistä kahdeksasta luokituksesta. Luokitusten perusteella voidaan jaotella myös ihmisryhmiä, joille tietynlaiset pelit ovat kiinnostavimpia. (Hunicke et al. 2004)

Farber listaa kirjassaan pelien neljäksi peruskomponentiksi Tavoitteen, säännöt, palautejärjestelmä ja vapaaehtoisen osallistuminen (Farber 2015). Optimaallinen pelillistetty järjestelmä sisältää näitä mahdollisimman suurissa määrin. Vapaaehtoisen osallistumisen osuus on kuitenkin oppimispainotteisessa pelillistetyssä ympäristössä hankala järjestää. Tuntemuksen vapaaehtoisuudesta voi kuitenkin saada aikaan tarpeeksi vaihtoehtoisia valintoja sisällyttämällä.

2.2 Pelillistämisen käsite ja periaatteet

Pelillistäminen on käsitteenä laaja-alainen ja jokseenkin vakiintumaton. Yleisimmin pelillistämisellä tarkoitetaan pelillisten tai leikillisten elementtien sisällyttämisestä kohteisiin, joissa niitä ei yleisemmin käytetä. Huotarin ja Hamarin mukaan tämä on ensimmäinen tunnettu määritelmä pelillistämiselle. He kuitenkin ovat ehdottaneet yleispäteväksi määritelmäksi: ”Palvelun parantamisen prosessi, jossa on pelillisten kokemuksien käyttömahdollisuuksia käyttäjän yleisen arvonluonnin tukemiseksi”. (Huotari & Hamari, 2012). Gilbert taas määrittelee pelillistetyn järjestelmän ”Järjestelmä joka sisältää pelimenetelmiä tiedonluontiin käyttäen peliympäristöön riippumattomia tuloja ja lähtöjä”. (Gilbert, 2016)

Farber, Gilbert ja Malone ovat kaikki luetelleet omissa teksteissään peleille ja pelillistetyille järjestelmille tärkeitä ja olennaisia ominaisuuksia. Pelien ja pelillistettyjen järjestelmien tulisi luoda käyttäjää kiinnostava ja aktivoiva ympäristö. Selkeät, tavoitettavissa olevat maalit ja välietapit ovat tärkeä osa toimintaa ja siitä syntyvää palautetta.



Kuva 1. *Pelillistetyn järjestelmän arkkitehtuuri (Gilbert 2016, s.35)*

Avataan tätä sanoen, että pelillistämisen tarkoituksena on vedota ihmisen luontaisiin tarpeisiin ja haluihin ja niitä hyödyntäen herättää kiinnostusta ja saavuttamisen tunnetta pelillistettävään tapahtumaan.

2.3 Esimerkkejä pelillisen ympäristön toteuttamisesta

Keinoja pelillistämisen toteutukseen on olemassa lähes rajattomasti. Pelillistämistä toteutetaan aktiivisesti muun muassa kaupallisissa ja koulutuksellisissa ympäristöissä.

Pelillistämistä hyödynnetään paljon esimerkiksi kännykälle ladattavissa sovelluksissa, kuten muistikirjana ja urheilunseurantajärjestelmänä motivoimaan ja innostamaan suoritusten toteuttamisessa ja tavoitteiden saavuttamisessa.

Malone on tutkinut pelien ja siihen liittyvän opetuksen itseisarvoista kiinnostavuutta. Tutkimuksessaan hän toteaa kolmen osa-alueen olevan tärkeitä pelien, ja tässä yhteydessä tätä myötä niiden liittyvien oppien kiinnostavuuden kannalta. Nämä osa-alueet ovat haasteet, fantasiat ja uteliaisuus. Osa-alueiden todettiin tehdyssä tutkimuksessa lisäävän osallistujien kiinnostuneisuutta aihepiiriin. Tärkeiksi elementeiksi todettiin selkeä, tiivis tavoite, sopivantasoinen haastavuus, riittävän selkeä palaute ja uteliaisuutta herättävät elementit kuten satunnaisuus. (Malone 1980, s.3-5) Tutkimuksessa hän käsittelee ja vetoaa myös aiemmin tehtyihin tutkimukseen aihepiiriin liittyen.

Tämä tutkimus keskittyy pelillistämisen toteuttamiseen opetuksellisissa merkeissä, Pelillistämisen toimivuus opetuksen apuvälineenä on todettu useissa tutkimuksissa. Esimerkiksi Farber kertoo kirjassaan Gamify your classroom esimerkkejä pelillistämisen toteutumisesta käytännössä ja pohtii niiden vaikutuksia oppimistilanteisiin. (Farber 2015) Malone on suorittanut tutkimuksia alakoululaisten opintoihin liittyen siihen mikä tekee peleistä kiinnostavia ja motivoivia, ja mikä kannustaa oppimaan peleissä. (Malone 1980) Molemmista näistä löytyy useita hyviä esimerkkejä, joita aiheesta kiinnostuneet lukijat voivat tutkia tarkemmin.

2.4 Pelillistämisen sitominen projektiin

Tässä projektissa pelillistämistä on lähdetty toteuttamaan käytettyyn ohjelmointiympäristöön sidottavan visualisoinnin avustuksella, joka toteutetaan käyttäen soveltuvaa kehitysympäristöä. Tavoitteena on luoda pelillinen elementti, joka innostaa ja motivoi opiskelijoita toteuttamaan harjoitustyön koodia ja avustaa virhetilanteiden selvityksessä visuaalisen palautteen muodossa. Tavoitteena on vedota opiskelijoiden uteliaisuuteen ja tehtävän suoritushaluun palkiten visuaalisella palautteella onnistuneita suorituksia esimerkiksi auton ajaessa onnistuneesti sisään parkkihallin portista.

3 KÄYTETYT OHJELMISTOT

Kappaleessa esitellään lyhyesti sovelluksen toteuttamiseen käytettävissä olevia erilaisia kehitysympäristöjä. Vaihtoehdoista valittiin soveltuva ympäristö, jolla sovellusta lähdettiin toteuttamaan. Lisäksi eritellään muita työlle olennaisia ohjelmistoja ja käsitteitä.

3.1 PLC ja IEC 61131-3

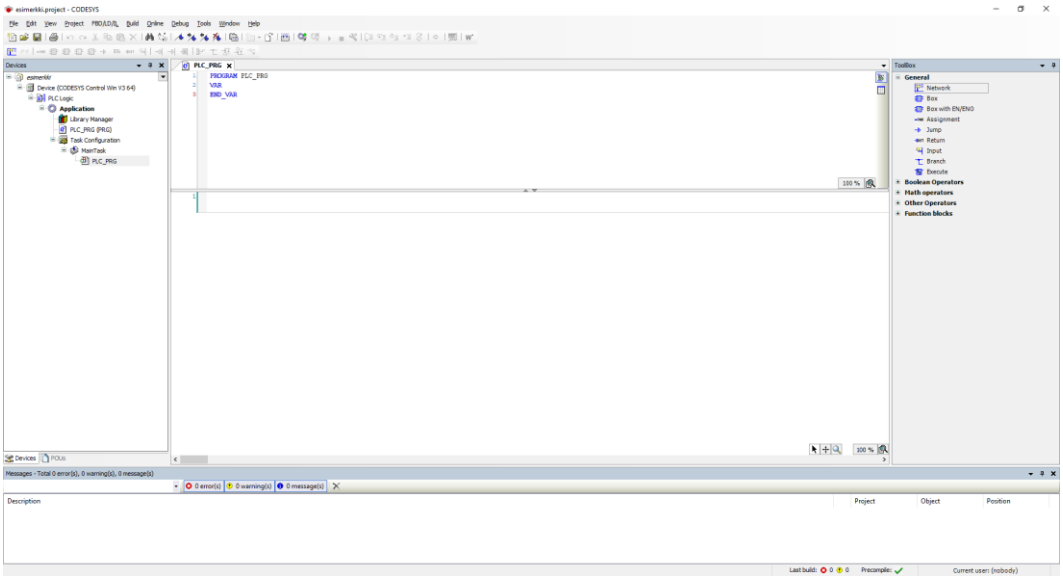
PLC, IEC61131-3 ja Codesys ovat harjoitustyön kannalta olennaisia käsitteitä. Tämän vuoksi niitä avataan hieman tässä kappaleessa. Myös ohjelmien sidostamista toisiinsa työn aikana kuvataan lyhyesti.

Programmable Logic Controller, Ohjelmoitava logiikkaohjain (PLC) tarkoittaa automatisoidun laitteiston ohjaamiseen suunniteltua tietokonetta. Yksinkertaisempi rakenne, korkeampi luotettavuus sekä halvempi hinta korvattaviin releisiin perustuviin järjestelmiin verrattuna. PLC sisältää ohjelmistolla määritettävissä olevia sisääntuloja ja ulosmenoja, jotka mahdollistavat monipuoliset käyttökohteet ja toiminnallisuuden. (Petruzella, 2005. s. 4)

Standardi IEC 61131-3 käsittelee ohjelmistoarkkitehtuuria ja ohjelmointikielet PLC-laitteistojen ohjelmointiin. Standardissa luetellaan viisi käytettyä ohjelmointikieltä ja määritellään ohjelmiston peruselementit. Standardin ensimmäinen versio julkaistiin 1983, ja se on päivitetty viimeksi vuonna 2013. (IEC 61131-3) Suurimpien laitevalmistajien PLC laitteistoissa käytetyt ohjelmistot pohjautuvat tämän standardin määrittelemiin perusteisiin vähintään perustavalla tasolla (Hannsen 2015, s. 134).

3.2 Codesys

Codesys on 3S-Smart Software Solutions GmbH:n vuonna 1994 luoma, standardin IEC 61131-3 mukaisen ohjelmointikielten toteuttamiseen tarkoitettu ohjelmisto. Ohjelmisto on saatavilla ilmaiseksi valmistajan verkkosivuilta. Codesys on maailmanlaajuisesti tunnettu useiden valmistajien käytössä oleva kehitysalusta. (3S-Smart Software Solutions GmbH. 2018) Esimerkkinä ABB:n Automation Builder sisältää Codesys:n ohjelmiston mukana (ABB 2018, s. 1). Tämän työn yhteydessä Codesys on kurssin harjoitustyön ohjelmointialusta PLC ohjelmointiin tutustuessa ja sen harjoittelussa. Näkymä uudessa, tyhjässä Codesys projektissa on nähtävissä kuvassa 2.



Kuva 2. *Uusi Codesys -projekti ja käyttöliittymä*

Codesysin etuja muihin vastaaviin ratkaisuihin ovat muun muassa hyvin läheinen IEC 61131-3 standardin noudattaminen, ohjainpiirin valmistajasta riippumattomuus, sekä PLC laitteiston simulointimahdollisuudet. (Hannsen 2015, s.353-354)

Uuden projektin luomisen Codesys-ympäristössä voi Hannsenin ohjeiden mukaisesti karkeasti jaotella seuraavanlaisesti:

- Käynnistetään Codesys ja määritellään uusi projekti
- Säädetään asetukset kohteena olevalle PLC laitteistolle
- Asetetaan kommunikointiasetukset [Ohjelma ja PLC]
- Lisätään tarvittavat kirjastot
- Määritetään uudet ohjelmiston rakenneyksiköt (Program Organizational Unit, POU)
- Ohjelman koodaaminen, kompilointi ja virheenkorjaus
- Määritellään erikoisdatatyypit
- Ohjelman lataaminen joko laitteeseen tai simulaatioon

(Hannsen 2015, s. 354-380)

Harjoitustyön yhteydessä opiskelijaryhmille annetaan valmiiksi muokattu asetustiedosto koodin toteuttamisen pohjaksi. Tässä on säädetty valmiiksi verkkomuuttujat (Network Variables, verkon yli päivitettävät muuttujat) ja asetukset ohjelmistojen väliseen kommunikaatioon. Järjestely takaa helpon aloittamisen ohjelmoinnin parissa ja säästää harjoitustyölle varattua rajallista aikaa.

3.3 Kehitysohjelmistovaihtoehdot

Pelinkehitykseen on olemassa useita ohjelmistoympäristöjä, joiden tarkoituksena on nopeuttaa ja helpottaa pelinkehityksen prosessia pelimoottorille, jonka yhteydessä kehitysympäristö tarjotaan. Jokaisella ohjelmistolla on hieman toisistaan poikkeava ohjelmien toteutustapa ja vaihtelevat tarjottujen ominaisuuksien puolesta. Erilaisia suosittuja pelinkehitysohjelmistoja ovat esimerkiksi Unreal engine, Gamemaker studio ja Unity 3D. (Cowan & Kapralos 2014) Lisäksi tässä työssä toteutettavan pelielementin/visualisaation toteuttaminen onnistuisi muuhunkin tarkoitukseen tarkoitettuja ohjelmistoja käyttäen. Esimerkkinä vaihtoehtoinen ratkaisu voitaisiin toteuttaa käyttäen Mevea-simulaatiota. Tässä työssä vertailu rajattiin näihin neljään vaihtoehtoon.

Jokainen mainituista ohjelmistoista pohjautuu johonkin ohjelmointikieleen ja omanlaiseen toimintarakenteeseen, jokaisen kehitysympäristön tarkoituksena on kuitenkin olla toimiva alusta monipuolisten sovellusten tai simulaatioiden toteuttamiseen. Lyhyt koonta kehitysympäristöistä ja niiden ominaisuuksista on liitteessä II.

Kolmen vertailtavan pelinkehitysympäristön ominaisuudet ovat jokseenkin samanlaiset: jokaisen tavoitteena on olla helposti käytettävä, laajan käyttäjäkunnan kattava kehitystyökalu, tämä näkyy muun muassa kehitysympäristön luontevassa vedä ja pudota tavassa luoda pelimaailmaa ja siinä miten usean eri alustan tuki, ja sitä myötä käyttäjäkunnan saavuttaminen, on helposti toteutettavissa. Jokaisesta näistä kolmesta ympäristöstä löytyy yhteisön markkinapaikka, jossa kehittäjät voivat jakaa ja myydä tekemiään pelielementtejä keskenään ja käyttää näitä omissa projekteissaan. Tämä vähentää pelinkehityksen työtaakkaa mahdollistaen keskittymisen esimerkiksi pelimekaniikkoihin yksittäisten pelimaailman esineiden mallintamisen sijasta.

Mevea eroaa luonteeltaan kolmesta aiemmin käsitellystä ympäristöstä. Mevea on suunniteltu teollisuuden simuloinnin työkaluksi viihdyttävien pelien luonnin sijaan. Tästä johtuen Mevea-simulaatio on vaihtoehtona haastavimmin toteutettavissa työn suunnitellussa käyttötarkoituksessa. Codesys-ympäristön kanssa kommunikointi olisi kuitenkin helpommin toteutettavissa valmiita ohjelmistoon integroidumpia rajapintoja käyttäen.

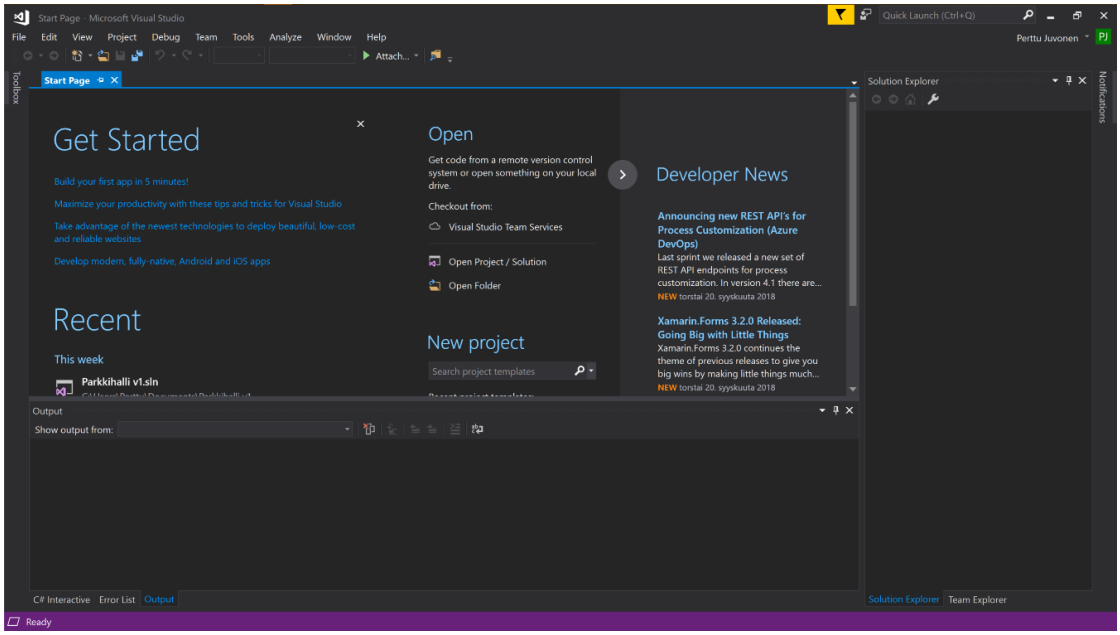
Gamemaker studio todettiin tähän projektiin sopimattomaksi maksullisuutensa ja valmiin Codesys-rajapinnan puutteen takia. Simulaation todettiin myöskin olevan paremmin

toteutettavissa kolmiulotteisessa ympäristössä. Rajapinnan puute sulki myös Unreal Enginen pois vaihtoehdoista, muiden ominaisuuksien soveltuessa hyvin tämän projektin tarpeisiin. Kehityksen helppouden vuoksi, ja saatavilla olevan CodeSysNetVars rajapinnan takia päädyttiin tuottamaan sovellus Unity 3D ympäristössä, Mevean vaatiessa enemmän työtä ympäristön mallien toteuttamisessa pelillisiin elementteihin keskittymisen sijasta yhteisön kauppapaikan puuttuessa.

3.4 Unity 3D ja Visual Studio

Työ päädyttiin toteuttamaan Unity 3D-kehitysympäristössä helpon saatavuuden, kattavan dokumentaation ja maksuttomuuden vuoksi. Lisäksi Unity 3D tarjoaa kattavan valikoiman muiden kehittäjien luomia esineitä ja ympäristöjä, hyödyllisiä kappaleita (asset), Unity Asset store-palvelussaan (Unity Tech. 2018). Kappaleet ovat ladattavissa joko ilmaiseksi tai maksua vastaan tekijän oman päätöksen mukaisesti, ja tämän jälkeen käytettävissä lataajan omassa projektissa.

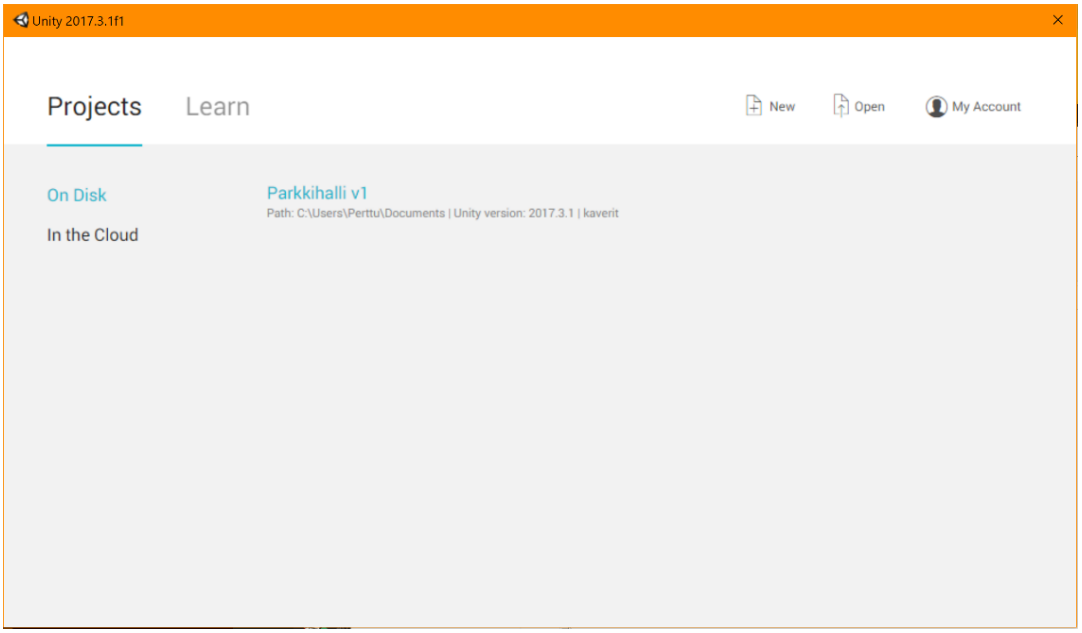
Unity 3D perustuu C#-ohjelmointikielellä tehtäviin skripteihin ja niiden integroimiseen visuaalisessa kehitysympäristössä tehtyihin komponentteihin. C# on Microsoftin vuonna 2002 julkaisema objektiorientoitunut ohjelmointikieli (Microsoft Corp. 2002, s. 1). Oletuksena Unity käyttää scriptien kehitysympäristönä Microsoftin Visual studio-kehitysympäristöä ja integroi koodin päivittämiset ja compiloinnit saumattomasti tämän kanssa. Tässä projektissa käytettiin Visual studion vuoden 2017 ilmaista yhteisöversiota (community edition), kehitysympäristön käytön jatkaminen vaatii Microsoft-tunnuksen 30 päivän mittaisen tunnuksettoman kokeilujakson jälkeen (Microsoft Corp. 2018). Microsoft Visual studion käyttöliittymä on nähtävissä kuvassa 3.



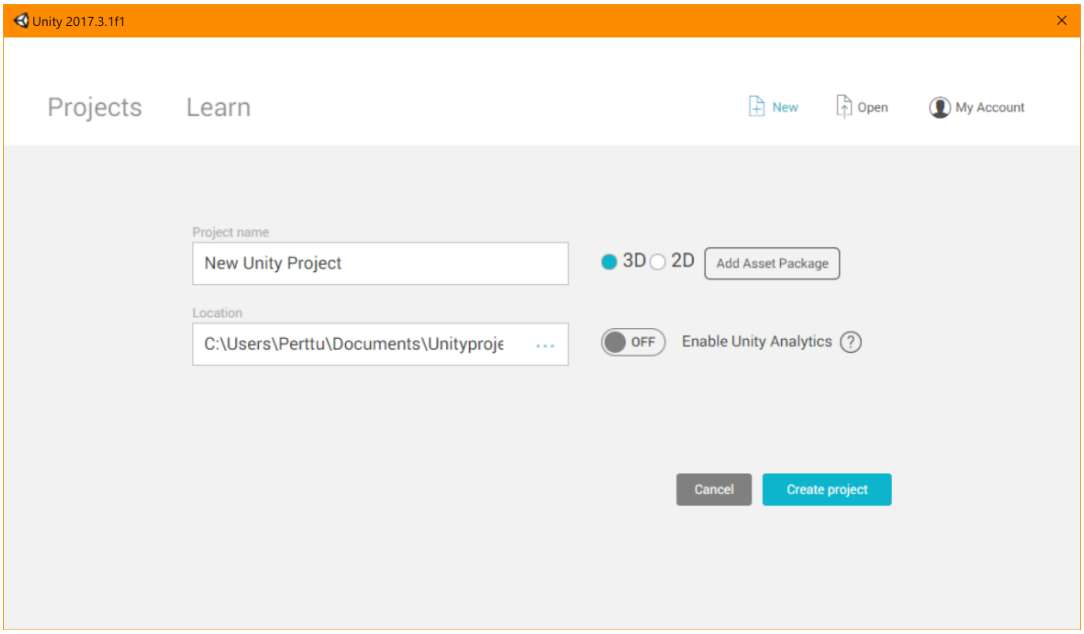
Kuva 3. Microsoft Visual Studion (2017 Community) käyttöliittymä

Työn toteutukseen käytetään Unityn kehitysalustan versiota 2017.3.1f1. Unity mahdollistaa valmiin ohjelmisto kohdentamisen joko selainpohjaiseksi (Webgl) tai Windows pohjaiseksi (.exe, executable) ohjelmaksi tarpeen mukaisesti. Muille alustoille ei tämän työn ohjelmaa tarvitse kehittää, vaikka niille ohjelman saisikin käytetyillä työkaluilla.

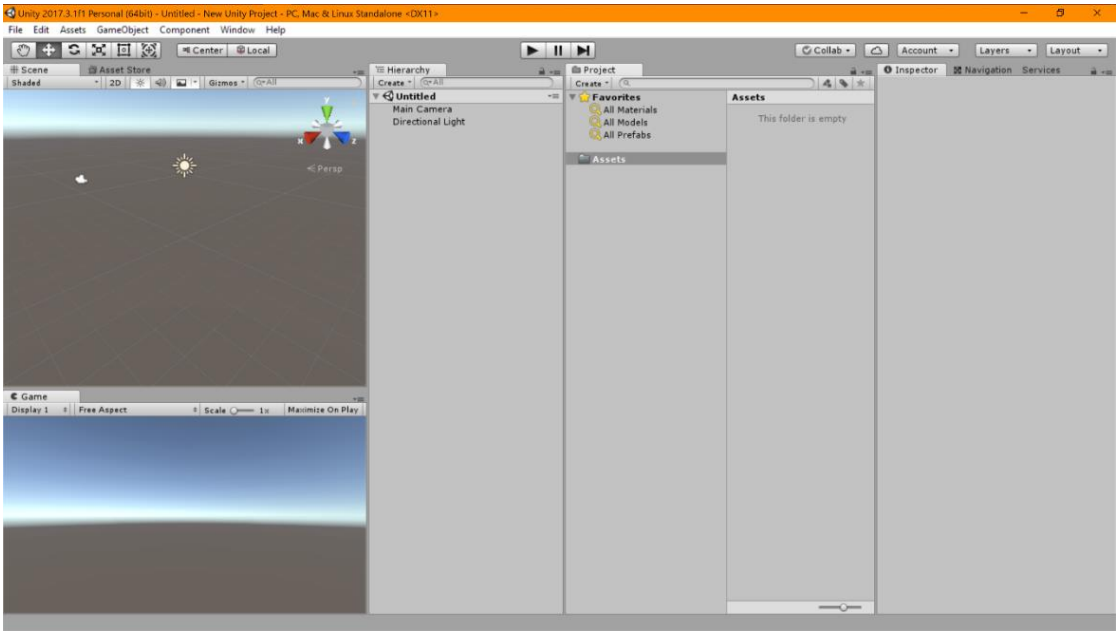
Uuden projektin luominen Unityssä on yksinkertaista, ohjelmistoa avatessa voi valita jatkavansa vanhaa projektia, tai luoda kokonaan uuden projektin. Unityn aloitusnäkymä ja uuden projektin luomisnäkymä ovat nähtävillä kuvissa 4 ja 5. Uuden projektin tyhjä ympäristö, johon projektia aletaan rakentamaan sekä Unityn yleiskäyttöliittymä näkyvät kuvassa 6.



Kuva 4. Unityn käynnistysnäkymä



Kuva 5. Uuden projektin luontinäkymä valintoineen



Kuva 6. Unity 2017r3 käyttöliittymä ja tyhjän uuden projektin ympäristö

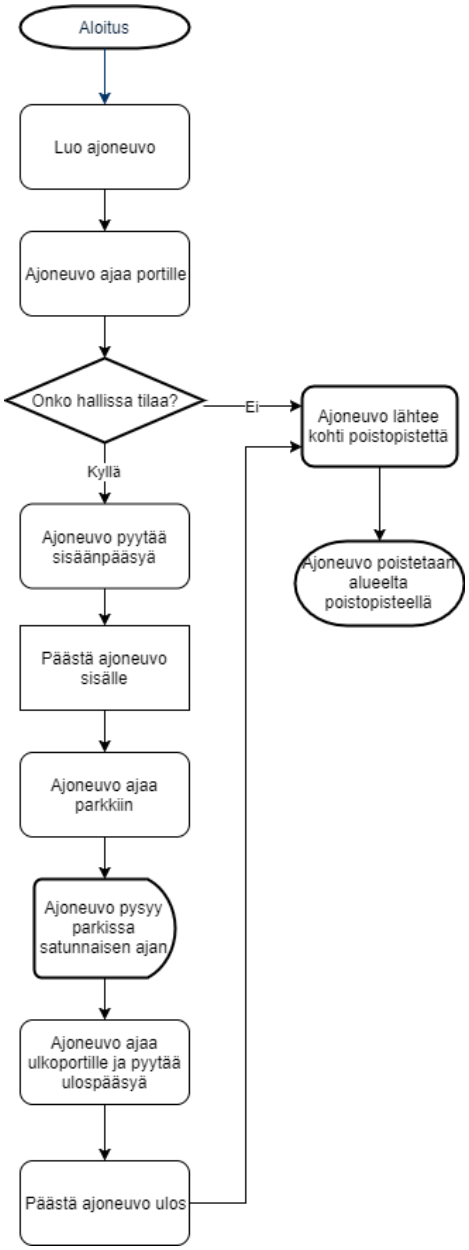
4 TUOTETTAVA OHJELMA

Kappaleessa täsmennetään tuotettavan ohjelman tarkoitusta, rakennetta ja toteutusmenetelmiä.

4.1 Ohjelman sisältö ja tarkoitus

Ohjelman perusrakenne ja sisältö rakentuvat vuoden 2016 mekatroniikan harjoitustyön ohjeistuksen pohjalle. Harjoitustyön tavoitteena on toteuttaa toimiva ohjausjärjestelmä parkkihallin loogiseen ohjaukseen. Työssä hyödynnetään ja harjoitellaan kurssilla aiemmin opittuja tietoja PLC-ohjelmointiin liittyen. Harjoitustyön vuoden 2016 ohjeistus on nähtävissä liitteessä I (Luostarinen 2016).

Ohjelmassa toteutetaan parkkihalli, jonka ohjausjärjestelmää opiskelijoiden tulee ohjata käyttäen ennaltamäärättyä, soveltuvaa laitteistoa. Laitteiston ohjelmisto toimii käyttäen PLC-ohjainta ja tämän toiminnot toteutetaan Codesys-ohjelmointiympäristössä. Laitteiston osia ovat parkkihallin portit, maksujärjestelmä, parkkeeraavien ajoneuvojen määrän hallinta, sekä parkkihallin tilasta, tai tarkemmin sen loppumisesta, ilmoittavat valot. Parkkihallin lisäksi tämän työn ohjelma luo ja käsittelee parkkihalliin parkkeeraavia ajoneuvoja, joiden käyttäytymiseen PLC-ohjelmoijilla ei ole vaikutusta. Ajoneuvot ovat niin sanottuja ei pelattavissa olevia hahmoja (NPC, non-playable characters tai non-player character) luodussa ympäristössä, jotka toimivat autonomisesti peliympäristössä ennaltamäärätyllä logiikalla. Autot pyrkivät sisälle parkkihalliin, parkkeeraamaa, ja poistumaan hallista sopivan ajan kuluttua. Ajoneuvon kulkuprosessi pääpiirteissään on kuvassa 7 olevassa kaaviossa.



Kuva 7. Ajoneuvon kulkeminen peliympäristössä

4.2 Ohjelman lähtökohdat

Työ toteutetaan käyttäen Unity-pelimoottoria ja moottorin ympärille rakennettua graafista kehitysympäristöä. Toteutettu sovellus kommunikoi kurssilla käytetyn Codesys-ohjelmointiympäristön kanssa soveltuvaa rajapintaa käyttäen. Työ toteutetaan valmiita Unityn Asset storesta saatavilla olevia pelikomponentteja käyttämällä, joten 3D-malleja ja niiden tekstuureita ei tämän työn ohessa ole mallinnettu. Taulukko käytetyistä asseteista, jotka eivät sisälly Unityn mukana toimitettuun assetpakettiin löytyy liitteestä IV.

Sovellusten välisenä rajapintana sovelletaan valmista UDP-verkkoprotokollaan (User Datagram Protocol) perustuvaa EasyNetVars/CodeSysNetVars -ohjelmistopakettia. EasyNetVars/CodeSysNetVars on GNU General Public Licence (GPL) -lisenssillä julkaistu, vapaassa jakelussa oleva rajapinta. GPL lisenssi sallii ohjelman käytön vapaasti ei-kaupallisissa tarkoituksissa (FSF 2007). Rajapinnalla kyetään siirtämään dataa Codesys- ja C#-ohjelmistojen välillä. Rajapinnan on kehittänyt Stefan Roßmann ja se on saatavissa GitHub-palvelusta. Linkki ohjelmiston sivustolle GitHub-palvelussa löytyy liitteestä III. Sivuilta löytyy myös esimerkkitoteutus rajapinnan käyttöönottoa varten.

Tuotetussa ohjelmistossa rajapinnan tehtävä on siirtää ohjauskomennot sovelluksen ja ohjelmointiympäristön välillä reaaliajassa hyödyntäen ohjelmistojen tukemaa UDP yhteyttä. Harjoitustyön tekijän toteuttaman koodin toiminta sidostuu visuaaliseen ympäristöön, joka myös simuloi tilannetta ympäristössä ja palauttaa tarvittavia tietoja harjoitustyön tekijän ohjelmalle. Muuttujat lähetetään ja vastaanotetaan Codesys:n puolella käyttäen verkkomuuttujalistaa (NVL, Network Variable List) ja järjestelmää joka on integroitu PLC laitteistojen ohjelmistoon. Kommunikaatio edellyttää laitekohtaista IP-osoitteiden ja verkkoporttien asettamista molemmissa ohjelmissa. Lisäksi molemmille verkkomuuttujalistoille täytyy määrittää oma listaidentifikaationumeronsa. Unity peliympäristössä kohteena olevan PLC:n IP:n asettaminen tapahtuu käyttöliittymän yläreunasta löytyvää tekstilaatikkoa käyttäen.

EasyNetVars/CodeSysNetVars on tuotu Unityssä luotuun ohjelmaan Visual studion kautta. Ohjelma on asetettu ohjelmiston riippuvuudeksi ja rajapintaa käsitellään skripteissä omassa nimiavaruudessaan. Ohjelmiston toiminnallisuuden takaavat asetukset, ja välitettävien muuttujien määrittäminen tapahtuvat ohjelmassa liitteestä V löytyvässä IOscript.cs-tiedostossa. EasyNetVars vaatii välitettävien verkkomuuttujien järjestyksen ja sisällön tuntemista ennalta, sillä muuttujat välittyvät yhtenä tietopakettina, jonka purkaminen tapahtuu ennaltamäärättyjä tietoja kuten muuttujien tyyppiä käyttäen. Muuttujatyypit määritellään ja kerätään listaan ennen paketin lähettämistä tai vastaanottamista, jonka jälkeen rajapinta pakkaa tai purkaa UDP-paketin jatkoäyttöä varten.

5 TULOKSET

Tässä kappaleessa esitellään tehdyn ohjelman valmistusvaiheet, kehitysprosessin aikaisia huomioita, sekä valmistettu ohjelma lopullisessa muodossaan.

5.1 Ohjelman toteutus

Ohjelman tekeminen aloitettiin suunnittelemalla peruselementit, joita sen tulisi sisältää. Suunnittelussa seurattiin pääasiassa kurssin harjoitustyön ohjeistusta, kuitenkin omaa näkemystä ja harkintaa käyttäen ominaisuuksien toteutuksessa. Ohjelmisto toteutetaan modulaarisesti ja skaalattavasti jotta sen sisältöä ja toimintaa pystytään muokkaamaan tarpeen mukaan. Koodeissa on myös pyritty noudattamaan hyvää selkeää ohjelmointityyliä englanninkielisiä muuttuja ja funktionimikkeitä sekä kommentointia käyttäen. Lista ohjelmistossa käytetyistä muiden valmistamista pelikomponenteista, 3D-malleista ja ohjelmistoista löytyy liitteestä III. Lisäksi tuotetut skriptit, niiden toiminnallisuus, sekä sijainnit peliympäristössä on taulukoitu liitteessä IV. Työn aikana toteutetut ja ympäristön lisätyt skriptit ovat lisäksi liitteinä V-XIX.

Kuva 8. *Peliympäristö kokonaisuudessaan*

Toteutus aloitettiin luomalla ympäristö. Ympäristönä toimii Unityn vakiokomponentteihin kuuluvalla maastonmuokkaukseen tarkoitetulla työkalulla luotu ruohotasanne, jota ympäröivät kukkulat. Ympäristöön etsittiin ja sijoitettiin Asset Storesta löytyvät parkkihalli sekä modulaarisista paloista koostuva tie, jota pitkin ajoneuvot liikkuvat. Käytetty parkkihalli on maksullinen tuote Unityn kauppapaikasta. Parkkihalliin ja sen ympäristöön

sijoitettiin näkyvien kohteiden lisäksi peliobjekteja toiminnallisuutta toteuttaville skripteille sekä ajoneuvojen liikkeitä rajoittaville ajoesteille. Myös ajoneuvojen liikkeitä ohjaavat kohteet, välitapit (waypoint), sijoitetaan ympäristöön. Näistä esimerkkinä mainittakoon parkkihallin neljään kerrokseen sijoitetut yksittäiset parkkipaikat, jotka sisältävät skriptin ajoneuvon parkkeeraamiskäyttäytymisen ohjaukseen sen saavuttua parkkiruutuun. Nämä objektit piilotetaan käyttäjien näkyvistä, sillä ne ovat tarpeellisia pelkästään toiminnallisuuden takaamiseksi. Kuvissa 9-11 nämä pelitilanteessa näkymättömät esteet ovat korostettu vihreillä raameilla.

Kuva 9. *Välietappeja peliympäristössä korostettuna vihrein raamein*

Kuva 10. *Toisen kerroksen parkkipaikkoja merkitsevät peliobjektit*

Kuva 11. *Ajoneuvojen liikettä rajoittavia esteitä*

Parkkihallin puomit asetettiin liikkumaan käyttäen Unityn niveltyökalua, tämän avulla puomeja pystytään parametria muuttamalla avaamaan ja sulkemaan. Puomien liikerata rajoitettiin 0-40 asteen alueelle, ja niihin lisättiin peliobjektit ääriasentojen saavuttamisen ilmaisemiseksi. Raja-anturit ovat osa harjoitustyön tehtävänantoa ja niiden tila välitetään järjestelmää ohjaavalle koodille. Puomien lisäksi hallin seinälle lisättiin merkkivalot, joiden tarkoitus on ilmaista parkkihallin käyttöastetta. Vihreä valo saatavilla olevia paikkoja ja punainen täyttä hallia. Valo näkyy kuvassa 6.

Kuva 12. *Hallin tilaa ilmaiseva valo portin viereisellä ulkoseinällä*

Valmiiseen ympäristöön luotiin ajoneuvoja, jotka pyrkivät parkkeeraamaan luotuun parkkihalliin. Ajoneuvojen ulkomuoto löytyy asset storesta pakettina yksinkertaisia automalleja. Mallin koko skaalattiin ympäristöön sopivaksi, alkuperäisen koon ollessa

suhteessa epärealistisen pieni. Ajoneuvoon sijoitettiin unityn sisältämä Navmesh agent, yksinkertainen keinoäly (AI, Artifical Intelligence), joka osaa navigoida sille määrättyyn kohteeseen luodussa peliympäristössä. Navmesh agent on suunniteltu pääasiassa humanoidityyppisten pelihahmojen liikutteluun, mutta soveltuu hyvin autojen ohjaukseen tässä yhteydessä liikkeiden todenmukaisuuden ollessa lähes merkityksettömät toteutetussa käyttötarkoituksessa. Ajoneuvoon lisättiin myös estekappale ja toiset autot aistiva puskuri, joiden avulla vältetään yhteentörmäyksiä ja parannetaan ajoneuvojen jonotuskäyttäytymistä puomeilla, esimerkki kuvassa 13.

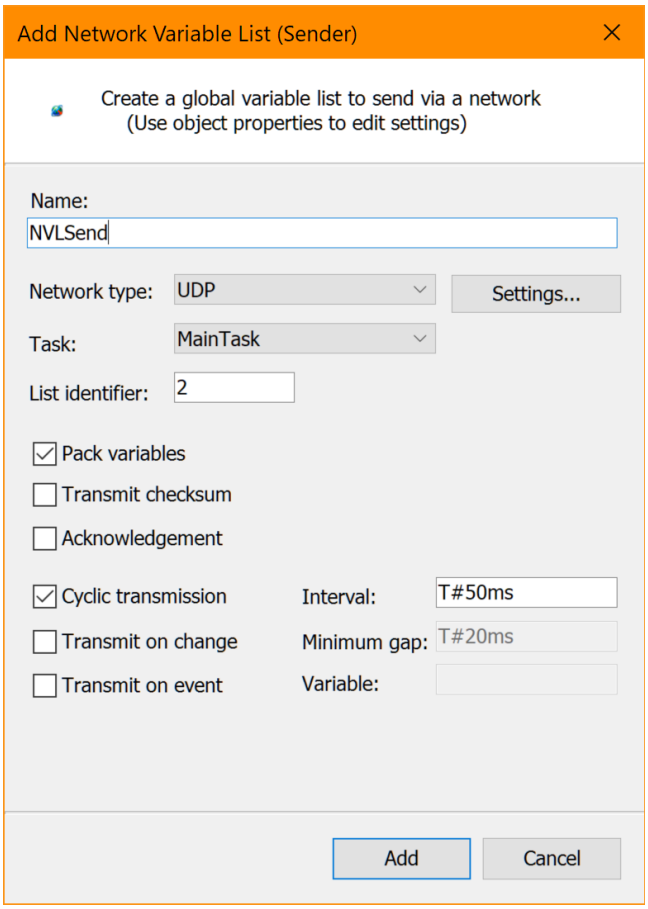
Kuva 13. *Ajoneuvot pyrkimässä halliin sisälle*

5.2 Valmiin ohjelman toiminta

Ohjelmisto pitää kirjaa pelialueella olevista ajoneuvoista, parkkipaikoista, ohjattavien komponenttien tiloista. Ohjelmistossa on rajattu suurin yhtäaikaisten ajoneuvojen määrä pelialueella.

Codesys-ympäristön ja ohjelman välillä jaetut muuttujat ovat pääasiallisesti boolean-tyyppisiä (totuusarvo). Ohjelmistojen saumaton toimiminta edellyttää muuttujalistojen järjestyksen olevan yhtenäinen kummassakin sovelluksessa, sillä lähetetyissä datapaketeissa ei muuttujien sisällöstä ole tietoja, vaan purkaminen perustuu ennaltamäärättyyn rakenteeseen. Muuttujalistat päivittyvät ohjelmasta toiseen tasaisin väliajoin, UDP-protokolla perustuu datan lähettämiseen ilman vastaanottotarkistuksia. Päivitykset toteutetaan 50ms välein ohjelman ollessa käynnissä riittävän vasteen takaamiseksi. CodeSysNetVars -rajapinnan luoma asetustiedosto, jonka voi tuoda suoraan Codesys:iin on

liitteessä XX. Tällä tiedostolla tuodaan Codesys:iin asetukset vastaanotettavien muuttujien tunnistamiseksi. Liitteessä XXI on esimerkki lähetettävien muuttujien listasta. Tälle listalle tulee asettaa listanumeroksi 2 ja verkkoportiksi jokin muu kuin 1202 muuten syntyvien päällekkäisyyksien takia. Tällä hetkellä tuotettu sovellus käyttää porttia 1203, mutta tämä on muutettavissa tarvittaessa. Codesys:in todettiin sallivan ainoastaan lähettävän listan verkkoportin muuttamisen, jolloin portti 1202 jää vastaanottavan listan käyttöön. Kuvassa 14. näkyy lähettävän nettimuuttujalistan asetussivu, portin ja IP-osoitteen säätö löytyy ”Settings” alavalikosta.



Kuva 14. *Lähettävän muuttujalistan asetussivu.*

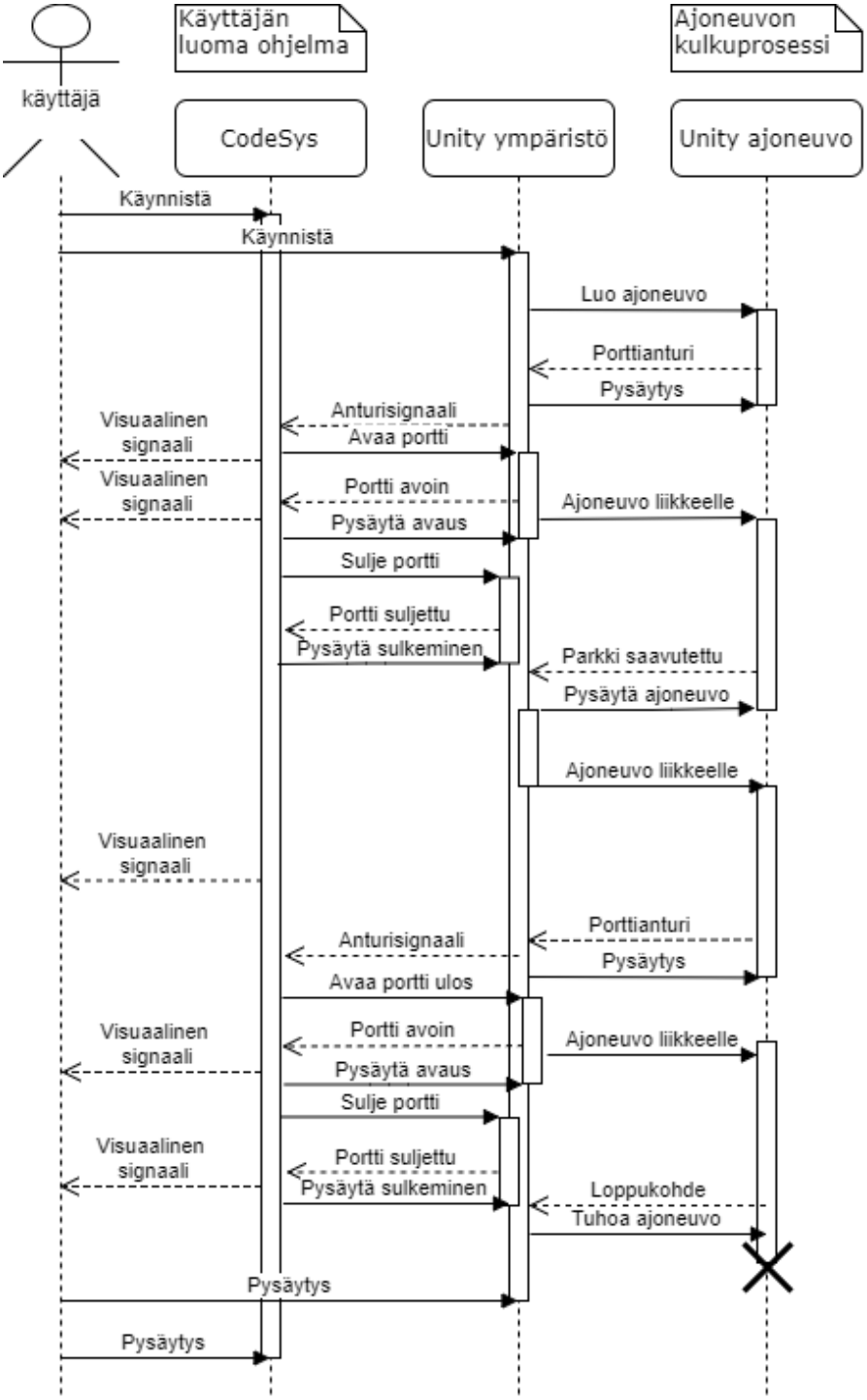
Ohjelman päänäkymä on kohdistettu parkkihallin puomeille. Näkymän yläosassa on alue ohjelman asetuksille ja toiminnoille, jossa määritetään ohjelmaan yhdistävän PLC:n IP-osoite ja voidaan käynnistää/tauottaa ohjelman toimintaa. Pelinäkymä esitetään kuvassa 15. Näkymässä on myös nappi ohjelman palauttamiselle oletustilaan. Ohjelman tauotus ja alkutilanteeseen palauttaminen toimivat myös näppäimistön p (pause) ja r (reset) näppäimiä painamalla ohjelman ollessa aktiivisena.

Kuva 15. *Näkymä ohjelman käyttäjälle*

Valmiissa ohjelmassa ajoneuvot ilmestyvät pelinäkymän ulkopuolelle tasaisin väliajoin ja ajavat parkkihallin portille, mikäli parkkihalli ilmaistaan olevan täynnä ohjausjärjestelmän toimesta, kääntyy ajoneuvo pois sisään pyrkimättä. Portilla olevasta autosta välitetään signaali ohjausjärjestelmälle, jonka seurauksena oletetaan järjestelmän avaavan portin saapuneelle ajoneuvolle. Kuvassa 16 näkyy parkkihalliin sisälle kulkemassa oleva ajoneuvo. Ajoneuvolle määritetään parkkipaikka satunnaisesti, joka annetaan ajoneuvoa ohjaavan agentin seuraavaksi kohteeksi. Varatuista parkkipaikoista ohjelmisto pitää kirjaa ja ei jaa näitä kohteeksi seuraaville ajoneuvoille ennen edellisen ajoneuvon poistumista. Ajoneuvo suunnistaa parkkipaikalle, jossa sille arvotaan pysäköintiaika. Pysäköintiajan täytyttyä parkkipaikan varaus poistetaan ja ajoneuvo suunnistaa parkkihallin uloskäynnin puomille, jossa ohjausjärjestelmälle välitetään signaali poistumassa olevasta ajoneuvosta, sekä tämän parkkilipukkeen maksutilasta. Portin auettua ajoneuvo saa kohteekseen pelinäkymän ulkopuolisen poistumispisteen, jossa se poistetaan pelialueelta. Samalla auto poistetaan myös pelialueella sijaitsevien ajoneuvojen listalta.

Kuva 16. *Hallin toimintaa porteilta käsin*

Ohjelman toiminnan yksinkertaistettu sekvenssikaavio löytyy kuvasta 17. Kaaviossa kuvataan harjoitustyön toimintaa ja eri ohjelmistojen osien välistä interaktiota ja prosessin oletettu kulkeutuminen opiskelijoiden tuottamassa harjoitustyössä. Osa ohjelmin sisäisistä ja välisistä toiminnoista on jätetty kaaviosta pois selkeyden ylläpitämiseksi.



Kuva 17. *Toteutetun ohjelman toiminnan yksinkertaistettu sekvenssikaavio*

6 JOHTOPÄÄTÖKSET JA YHTEENVETO

Kappaleessa kerrataan ja esitetään tutkimuksen aikana ilmenneitä aiheeseen liittyviä pohdintoja.

6.1 Pelillistäminen osana opetusta

Pelillistäminen on työkalu opetuksen apuna. Pelillistäminen ei korvaa ohjauksen tarvetta opetustapahtumassa, vaan auttaa opettajaa neuvomaan ja ohjeistamaan oppilaitaan tehokkaammin harjoituksen toteuttamisessa. Lisäksi pelillistämisen tulisi avustaa oppilaita oivaltamaan ja tutkimaan haluttuun aihepiiriin liittyviä tietoja ja taitoja. Koskinen, Kangas ja Krokfors toteavat kirjallisuuskatsauksessaan opettajan roolin olevan tärkeä pelillisen opetuksen suunnittelussa, toteutuksessa ja arvioinnissa käytetystä pelillisestä ympäristöstä riippumatta. Opettajan rooli vaihtelee pedagogisen prosessin suunnittelemisesta harjoituksenaikaiseen ohjaamiseen. (Koskinen et al. 2014, s. 27-34)

6.2 Tuotetun ohjelman toimivuus

Työssä tuotettu ohjelma on hyvä lähtökohta pelillistämisen hyödyntämiseen PLC-ohjelmoinnin opetuksessa. Ohjelma on käytännössä nykyisellään vain prototyyppi jonka kehittämistä tulisi jatkaa ensimmäisten käyttökokemusten perusteella. Tämän työn yhteydessä ei ohjelmiston toimintaa lähdetty yksinkertaista perustoiminnallisuuden testaamista syvällisemmin toteuttamaan. Ohjeman olennaisimmat toiminnot kuitenkin todettiin toimiviksi. Ohjelmisto tarjoaa perinteistä harjoitusta enemmän palautetta ja kannustaa kokeiluihin perustuvaa oppimista.

Ohjelmiston toteuttamisen yhteydessä kävi ilmi, että monissa kohdin toteutetun koodin rakenne voitaisiin toteuttaa tehokkaamminkin. Nämä epäkohdat olivat seurausta työn tekijän vähäisestä kokemuksesta käytetyn ohjelmointikielen kanssa. Yleiseen toimivuuteen näillä koodin parantamismahdollisuuksilla ei kuitenkaan ole suurta vaikutusta ohjelmiston suppeasta skaalasta johtuen. Optimisaation tarve lisääntyy ohjelmiston laajuuden mukana.

6.3 Ohjelman ja pelillistämisen kehitys jatkossa

Toteutetun sovelluksen pohjalta kurssin opetuksen pelillistämistä voidaan jatkaa tulevaisuudessa, myös muille kursseille menettelyn soveltaminen on mahdollisuus. Ohjelmistoon voidaan lisätä esimerkiksi kattavampi palaute- ja kannustamisjärjestelmä pisteytyksen muodossa, erilaisia tasoja tai ongelmatilanteita järjestelmässä, haasteita tai

esimerkiksi leaderboard: lista parhaiten tehtävästä suoriutuneista ryhmistä tai henkilöistä. Ohjelmiston ulkoasua ja persoonallisuutta voitaisiin myös kehittää esimerkiksi erityyppisten autojen ja näille tyypillisten käyttäytymismallien muodossa. Tämänkaltaisilla ominaisuuksilla sovelluksen kiinnostavuutta ja innostavuutta voidaan kehittää eteenpäin. Ohjelmiston vaikutuksesta oppimisen työkaluna tulisi pyytää palautetta harjoitustyötä tekeviltä opiskelijoilta ohjelmiston soveltuvuuden toteamiseksi ja ominaisuuksien kehittämiseksi.

Sovellusta voidaan myös soveltaa tosielämän tilanteiden mallintamiseen. Käytännössä tämä toteutuisi automatisoitavan kohteen luomalla Unityssä ja tämän pohjalle automaatiojärjestelmän toteuttamalla ja testaamalla. Tämänkaltainen mallintaminen kuitenkin vaatii halutun ympäristön replikoimisen riittävällä tarkkuudella, jonka tekemiseksi vaadittaisiin huomattava määrä työtunteja. Joissain tapauksissa tämä kuitenkin edistäisi automaatiojärjestelmän suunnittelua ja toteutusta mahdollistaen useiden ratkaisujen kokeilemisen ja vertailun.

6.4 Yhteenveto

Työssä esitettiin tiiviisti pelien ja pelillistämisen perusperiaatteita ja toteutusta käytännön tilanteissa, näitä mukaillen toteutettiin ohjelma Lappeenrannan teknillisessä yliopistossa järjestettävän mekatroniikan kurssin harjoitustyön avuksi. Harjoitustyössä opiskelijat luovat ryhmittäin PLC-laitteiston periaatteita ja Codesys ohjelmointiympäristöä käyttäen parkkihallin ohjausjärjestelmän. Luotu ohjelma toimii visuaalisena palautteena ja mielenkiinnon herättäjänä opiskelijoille harjoitustyön ohessa. Ohjelmiston toteuttamisen periaatteita ja mahdollisia kehitysalustoja eriteltiin ja erittelyn perusteella valittiin ohjelmiston kehitysympäristöksi Unity 3D. Unity 3D ohjelmiston ohessa kehityksessä käytettiin Microsoft Visual Studio 2017 ohjelmointiympäristöä. Codesys:n ja luodun ohjelmiston rajapintana sovelletaan valmista UDP-protokollaan perustuvaa rajapintaa. Toteutetun ohjelman teon vaiheet ja toteutunut ohjelma dokumentoidaan ja esitellään sen käyttöä ja jatkokehitystä tukevin perustein. Ohjelman jatkokehitykselle ja muille käyttökohteille esitetään vaihtoehtoja ja ehdotuksia työn aikana muodostuneiden havaintojen ja kokemuksen pohjalta.

LÄHDELUETTELO

ABB – Welcome to ABB Automation Builder 2.1.2. [ABB:n Automation Builderin jukaisutiedot]. ABB, 2018. [Viitattu 9.10.2018] Saatavilla: http://dg8gvgfk7mhsg.cloudfront.net/AB_ReleaseNotes/Automation_Builder_2.1/ReadMe.pdf

CODESYS - The comprehensive software suite for automation technology [Codesys-ohjelmiston virallisilla verkkosivuilla]. 3S-Smart Software Solutions GmbH, 2018. [Viitattu 6.5.2018]. Saatavissa: https://www.codesys.com/the-system.html.

Cowan, B. Kapralos, B. A Survey of Frameworks and Game Engines for Serious Game Development. 2014. Teoksessa: IEEE 14th International Conference on Advanced Learning Technologies. IEEE. 2014. s. 662-664.

Dombrowski, M. Lochner, J. Gamification of Modern Society: Digital Media's Influence on Current Social Practices. 2015. Teoksessa: Lee, N. Encyclopedia of Computer Graphics and Games. s. 334- 338. Springer International Publishing. Sveitsi. 2015.

Epic Games. Unreal Engine 4. [Epic Gamesin virallisilla verkkosivuilla]. Epic Games Inc. 2018. [Viitattu 4.8.2018]. Saatavissa: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

Farber, Matthew. Gamify Your Classroom – A Field Guide to Game Based Learning. 2015. Peter Lang publishing Inc., Yhdysvallat: New York.

Gilbert, Sari. Designing Gamified Systems – Meaningful Play in Interactive Entertainment, Marketing and Education. 2016. Focal Press, Taylor & Francis Group. Iso-Britannia: Oxfordshire.

FSF (Free Software Foundation Inc.). GNU General Public License – Version 3. 2007. [Verkkodokumentti]. [Viitattu 18.10.2018]. Saatavissa: https://www.gnu.org/licenses/gpl-3.0.en.html.

Hannsen, D. H. Programmable Logic Controllers – A Practical Approach to IEC 61131-3 Using Codesys. 2015. John Wiley & Sons, Ltd. Iso-Britannia: West Sussex.

Hunicke, R. LeBlanc, M. Zubek, R. MDA: A Formal Approach to Game Design and Game Research. 2004. AAAI Workshop - Technical Report. 1.

Huotari, K. Hamari, J. Defining Gamification - A Service Marketing Perspective. 2012. Julkaisussa: Proceeding of the 16th International Academic MindTrek Conference. MindTrek '12. ACM. Yhdysvallat: New York. s.17-22.

IEC 61131-3. 2013. Programmable controllers - Part 3: Programming languages. 3rd Edition. Geneve: International Organization for Standardization.

Koskinen, A. Kangas, M. Krokfors, L. 2014. Oppimispelien tutkimus pedagogisesta näkökulmasta. Teoksessa: Krokfors, L. Kangas, M. Kopisto, K. Oppiminen Pelissä – Pelit, pelillisyys ja leikillisyys opetuksessa. Vastapaino. Suomi: Tampere. 2014.

Luostarinen, Lauri. Harjoitustyö 1 | Logiikkaohjelmointi. 2016. Lappeenrannan Teknillisen Yliopiston, Mekatroniikan kurssin Moodle-ympäristö. [Ei julkisesti saatavilla].

Malone, H.W. What Makes Things Fun to Learn? – A Study of Instrictically Motivating Computer Games. 1980. Xerox Palo Alto tutkimuslaitos. Yhdysvallat: Kalifornia. 2005.

Microsoft Corp. Sign in to Visual Studio [Visual Studion verkkodokumentaatio] Microsoft Corp. 2018 [Viitattu 10.8.2018]. Saatavissa: https://docs.microsoft.com/en-us/visualstudio/ide/signing-in-to-visual-studio?view=vs-2017

Microsoft Corp. C# Language Specification – Version 1.0. 2002. Microsoft Corp. 2002.

Petruzella, F. D. Programmable Logic Controllers. Third Edition. 2005. McGraw Hill. Yhdysvallat: New York.

Unity Tech. Unity User Manual (2017.3). [Unity-kehitysympäristön digitaalinen dokumentaatio]. Unity Technologies. 2018. [Viitattu 10.8.2018]. Saatavilla: https://docs.unity3d.com/2017.3/Documentation/Manual/index.html

YoYo Games. Gamemaker Studio 2 [YoYo Gamesin virallisilla verkkosivuilla]. YoYo Games, Ltd. 2018. [Viitattu 4.8.2018] Saatavissa: https://www.yoyogames.com/gamemaker/features

LIITTEET

LIITE I: Vuoden 2016 mekatroniikan harjoitustyön ohjeistus

BK60A0200 MEKATRONIIKKA, SYKSY 2016

Open your mind. LUT.
Lappeenranta University of Technology

HARJOITUSTYÖ 1 | LOGIIKKAOHJELMOINTI

Harjoitustyö tehdään 4-5 hengen ryhmissä. Jokaisen jäsenen pitää pystyä vastaamaan työhön liittyviin
yksityiskohtaisiin kysymyksiin ja tekemään pieniä muutoksia pyydettäessä.

KIRJALLISIA TEHTÄVIÄ

Vastaa seuraaviin kysymyksiin. Vastauksen enimmäispituus on kymmenen riviä tekstiä / kysymys. Asian havainnollistaminen kuvilla on suotavaa.

1. Mikä on PLC? Mitkä ovat sen ominaispiirteet?
2. Mikä on releen toimintaperiaate?
3. Minkälaisen ohjaustekniikan logiikkaohjaimet ovat korvanneet?

OHJELMOI PYSÄKÖINTITALON OHJAUSJÄRJESTELMÄ

Ohjausjärjestelmään kuuluu seuraavat toiminnot

1. Puomien ohjaus ja autojen laskenta
    a. Kun auto tulee sisäänajoon, kuljettaja painaa nappia (, tulostetaan tunnistelappu) ja puomi avataan. Puomi sulkeutuu kun auto on ajanut sisään. Ajastin estää puomia sulkeutumasta välittömästi napin painalluksen jälkeen ja sulkee puomin, jos auto ei tulekaan sisään. Optisella digitaalianturilla mitataan milloin auto on poistunut puomin alta. Puomia ohjataan digitaalisella lähdöllä.
    b. Auton poistuessa parkkitalosta maksuautomaatti varmistaa, että maksu on suoritettu ja lähettää sen jälkeen digitaalisignaalin puominohjaukselle. Maksu automaatin toimintaa voidaan simuloida esim. käyttöliittymään tehdyllä napilla. Ei tarvitse ohjelmoida maksuautomaattia.
    c. Parkkitalossa olevien autojen lukumäärä täytyy laskea. Parkkitalon ulkopuolella näytetään merkkivalolla onko talossa tilaa vai ei. Näitä kahta merkkivaloa ohjataan digitaalisilla lähdöillä.

d. (Bonustehtävä: Käytä rajakytkimiä, jotka varmistavat ovatko puomit ylä- tai ala-asennoissa. Jos puomi jää jumiin keskellä, järjestelmä menee vikatilaa, josta ilmoitetaan valvomoon.)

2. Pysäköintitaloa valvotaan graafisen käyttöliittymän kautta, jossa on seuraavat toiminnot:

a. Näytetään

i. puomien asennot

ii. sisällä olevien autojen lukumäärä ja vapaiden paikkojen lukumäärä.

b. Ohjausmahdollisuudet

i. Sisäänajopuomin avaus ja sulkeminen

ii. Ulosajopuomin avaus ja sulkeminen

iii. Manuaalisesti lisätään ja vähennetään autolaskurin arvoa.

iv. Nollataan autolaskurin arvo.

LIITE II: Koonta ohjelmistojen ominaisuuksista

Ohjelma	**Kuvaus**	**Ohjelmointikieli**	**Tärkeimmät tuetut alustat**	**Erikoisominaisuudet**	**Hinta**	**Kauppaelementeille**	**Esimerkkejä toteutetuista projekteista**
YoYo Games Gamemaker Studio 2	2D -pelimoottori ja kehitysympäristö	GameMaker Language (GML)	Windows Mac OS Linux HTML5/ WebGL Mobiili-/Konsolialustat	Tuki ohjelmistolaajennoksille Vedä ja pudota kehitystyyli	Skaalautuu haluttujen ominaisuuksien mukaan 30-1500$/v	Kyllä, Marketplace	Toby Fox, Undertale
Unity 3D	2D ja 3D pelimoottori ja pelinkehitysympäristö	C#	Windows Mac OS Linux HTML5/WebGL Mobiili-/Konsolialustat	VR-tuki Vedä ja pudota kehitystyyli Pilvikehitysintegraatio	Skaalautuu kehittäjän tuottojen/haluttujen ominaisuuksien mukaan 0-125$/kk	Kyllä, Unity Asset Store	Colossal order, Cities Skylines StudioMDHR, Cuphead

Epic Games Unreal engine 4	3D- pelimoottori ja kehitysympäristö, käytetty myös markkinointivisualisaatioissa	C++	Windows Mac OS Linux Mobiili-/konsolialustat	Tuki ohjelmistolaajennoksille VR-tuki	Ilmainen (lisensointimahdollisuus)	Kyllä, Unreal Marketplace	Epic Games, Unreal Tournament 4 Digital Arrow/Nordic Games, Ark: Survival Evolved
Mevea Mevea Modeller /Solver	Alusta teollisuuden reaaliaikaisille 3D-simulaatioille ja tuotetestaukselle	C++	Windows	Sisäänrakennettu tuki ajoneuvo-ohjaamosimulaatiolle Keskittynyt simulaatiotarkkuuteen Simulink -yhteensopivuus	Erillissopimus	Ei	Mevea, Erinäiset työkonesimulaatiot

LIITE III: Ohjelman osat ja niiden toiminnallisuus

Skripti	Sijainti	Kuvaus
IOScript.cs	IO-Objekti	Hallinnoi sovelluksen ja Codesys ympäristön välistä rajapintaa/muuttujia sekä peliympäristön hallintaa
CarManager.cs	IO-Objekti	Hallinnoi ajoneuvojen sijaintia ja ohjaamista peliympäristössä, sisältää tietokannat ajoneuvoista ja parkkipaikoista sekä niiden tilasta
RestartS.cs	IO-objekti	Sisältää toiminnot ohjelman resetoimiselle/tauottamiselle
CarOnGate.cs	Sisäänmenoportin laukaisuobjekti	Hallinnoi portille saapuneen ajoneuvon tiedottamisen ja informaatiovälityksen ympäristön ja ajoneuvon välillä
CarOnGateOut.cs	Ulosmenoportin laukaisuobjekti	Hallinnoi portille saapuneen ajoneuvon tiedottamisen ja informaatiovälityksen ympäristön ja ajoneuvon välillä
GateMotorIn.cs	Sisäänmenoportin puomi	Puomin noston ja laskun toteuttavan moottorin ohjaus
GateMotorOut.cs	Ulosmenoportin puomi	Puomin noston ja laskun toteuttavan moottorin ohjaus
GinOpenTS.cs	Sisäänmenoportin puomin yläasento laukaisuobjekti	Välittää signaalin IO-skriptille sisäänmenopuomin ollessa auki
GinCloseTS.cs	Sisäänmenoportin puomin ala-asento laukaisuobjekti	Välittää signaalin IO-skriptille sisäänmenopuomin ollessa kiinni
GoutOpenTS.cs	Ulosmenoportin puomin yläasento laukaisuobjekti	Välittää signaalin IO-skriptille ulosmenopuomin ollessa auki
GoutCloseTS.cs	Ulosmenoportin puomin ala-asento laukaisuobjekti	Välittää signaalin IO-skriptille ulosmenopuomin ollessa kiinni
CarBumper.cs	Jokainen ajoneuvo	Tiedottaa ajoneuvon AI:lle lähestyvästä esteestä, mahdollistaa siistin jonottamisen
AutoAI.cs	Jokainen ajoneuvo	Hallinnoi yksittäisen ajoneuvon ohjaamisen määrättyyn kohteeseen, sisältää ajoneuvon tiedot
CarParked.cs	Jokainen parkkipaikka	Hallinnoi ajoneuvon parkkeeraamisen ja toiminnan hallin sisällä, välittää sijaintitietoa ympäristölle
Indicator_light.cs	Hallin tilaindikaattorivalo	Hallinnoi parkkihallin tilaindikaattorin toimintaa
DestroyCar.cs	Ajoneuvojen viimeinen välietappi	Tuhoaa/poistaa pelimaailmasta parkkihallista poistuneen ajoneuvon

LIITE IV: Muualta hankittujen osien lähteet ja tekijät

Ladattu malli/ohjelmisto/paketti	**Tekijä**	**Käyttötarkoitus**	**Hinta**	**Saatavilla**
Parking Garage – Complete	Polygon Land	Parkkihallin 3D-malli, ympäristö	4,38 €	Unity Asset store: https://assetstore.unity.com/packages/3d/environments/urban/parking-garage-complete-104909
Simple Modular Road Kit	Blacklight	Tiet, ympäristö	Ilmainen	Tuki loppunut, ei enää saatavilla
Simple Cars Pack	Myxerman	Ajoneuvojen 3D-mallit	Ilmainen	Unity Assrt Store: https://assetstore.unity.com/packages/3d/vehicles/land/simple-cars-pack-97669
GE 2-in-1 Canadian Pedestrian signal	LED signal guy	Hallin tilaindikaattorivalon kuori	ilmainen	3D Warehouse: https://3dwarehouse.sketchup.com/model/728fc165c84877136a9f2f356283872d/GE-2-in-1-Canadian-Pedestrian-signal
EasyNetVars/CodesysNetVars	Stefan Roßmann	Rajapinta Codesys muuttujille	ilmainen	GitHub: https://github.com/rossmann-engineering/EasyNetVars

LIITE V: IOScript.cs

```
using UnityEngine;
using EasyNetVars;
using System.Collections;


public class IOScript : MonoBehaviour //script handling all the IO
{

    NetVars srCodesysWriteVars = null;
    NetVars srCodesysNetVars = null;
    CDataTypeCollection collect;

    private bool HasError = false;
    private bool ErrorShown = false;

    //Settings for communication with Codesys
    public string ConnectionIP; //read from UI, default to localhost
    public int OutConnectionPort = 1202;
    public int InConnectionPort = 1203;

    //Controls for gate in
    public bool CarOnGateIn;

    public bool GInIsOpen;
    public bool GInIsClosed;
    public bool OpenGIn;
    public bool CloseGIn;

    public bool GetTicket; //Ticket button for a customer
    public bool CarLetIn;

    //Controls for gate out
    public bool CarOnGateOut;

    public bool GOutIsOpen;
    public bool GOutIsClosed;
    public bool OpenGOut;
    public bool CloseGOut;

    public bool PaymentComplete; //Is the payment of the car on the gate completed
    public bool CarLetOut;

    //Other controls
    public int floorsactive;
    public bool HallIsFull; //All parking places are full
    public bool HallHasRoom;  //There is parking places available
    public int CarsThrough; //Amount of cars gone through the process

    public NetVars SRCodesysNetVars;
    public bool generateGVL; //Generate the global network variable list for codes
ys

    //public int parkamount; //Amount of places in the parking hall

    private void OnGUI()
    {
        if (HasError && !ErrorShown)
        {
```

```
            ErrorShown = true;
            StartCoroutine(ShowError());
        }
    }

    private void Start()
    {
        Time.timeScale = 0f; //Start the game paused
        ConnectionIP = "127.0.0.1"; //set defaut IP
    }

    public void ToggleNetVars()
    {
            if (Time.timeScale == 1f)
            {
                InvokeRepeating("WriteVars", 0.0f, 0.05f); //send variables to cod
esys every 50ms
                InvokeRepeating("ReadVars", 0.0f,0.05f);
                Debug.Log("Variable connection running");
            }

            else
            {
                CancelInvoke("ReadVars"); //stop updating variable lists when paus
ed
                CancelInvoke("WriteVars");
            }
    }

    public void ReadVars() //Read  new values from codesys
    {
        srCodesysNetVars = new NetVars();

        srCodesysNetVars.CobID = 2; // should be the same as the list indentifier
in codesys

        srCodesysNetVars.IPAdress = ConnectionIP;
        srCodesysNetVars.Port = InConnectionPort;

        //Here is list of datatypes that will be read, Codesys GVL must have these
 in the same order!
        srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetVar
s.DataTypes.booltype)); //OpenGIn
        srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetVar
s.DataTypes.booltype)); //CloseGIn
        srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetVar
s.DataTypes.booltype)); //OpenGOut
        srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetVar
s.DataTypes.booltype)); //CloseGOut
        srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetVar
s.DataTypes.booltype)); //HallIsFull
        //srCodesysNetVars.dataTypeCollection.Add(new CDataTypeCollection(EasyNetV
ars.DataTypes.booltype)); //HallHasRoom

        try
            {
                //Debug.Log("Array try");
```

```
                ArrayList newValues = SRCodesysNetVars.ReadValues(); //read values
 from codesys
                //Debug.Log("Array get");

                if (newValues != null)
                {
                    OpenGIn = (bool)newValues[0];
                    CloseGIn = (bool)newValues[1];
                    OpenGOut = (bool)newValues[2];
                    CloseGOut = (bool)newValues[3];
                    HallIsFull = (bool)newValues[4];
                    HallHasRoom = (bool)newValues[5];
                }
            }

            catch
            {
                HasError = true;
            }
    }

    public void WriteVars() //collect and send new values from program
    {
        srCodesysWriteVars = new NetVars();

        srCodesysWriteVars.CobID = 1; // should be the same as the list indentifie
r in codesys

        srCodesysWriteVars.IPAdress = ConnectionIP;
        srCodesysWriteVars.Port = OutConnectionPort;

        //Here is list of datatypes that will be written, GVL order important
        collect = new CDataTypeCollection(CarOnGateIn, EasyNetVars.DataTypes.boolt
ype);
        collect.VariableName = "CarOnGateIn";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //CarOnGateIn

        collect = new CDataTypeCollection(GInIsOpen, EasyNetVars.DataTypes.booltyp
e);
        collect.VariableName = "GateInOpen";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GInIsOpen

        collect = new CDataTypeCollection(GInIsClosed, EasyNetVars.DataTypes.boolt
ype);
        collect.VariableName = "GateInClosed";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GInIsClosed

        collect = new CDataTypeCollection(CarOnGateOut, EasyNetVars.DataTypes.bool
type);
        collect.VariableName = "CarOnGateOut";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //CarOnGateOut

        collect = new CDataTypeCollection(GOutIsOpen, EasyNetVars.DataTypes.boolty
pe);
        collect.VariableName = "GateOutOpen";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GOutIsOpen
```

```
collect = new CDataTypeCollection(GOutIsClosed, EasyNetVars.DataTypes.booltype);
        collect.VariableName = "GateOutClosed";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GOutIsClosed

        collect = new CDataTypeCollection(CarLetIn, EasyNetVars.DataTypes.booltype
);
        collect.VariableName = "CarLetIn";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GOutIsOpen

        collect = new CDataTypeCollection(CarLetOut, EasyNetVars.DataTypes.booltyp
e);
        collect.VariableName = "CarLetOut";
        srCodesysWriteVars.dataTypeCollection.Add(collect); //GOutIsClosed

        if (generateGVL)
        {
            srCodesysWriteVars.CreateGVLFile("C:\\Users\\Perttu\\Desktop\\GVLFile.
GVL"); // this line is for generating the variable list for codesys
            generateGVL = false;
        }

        try
        {
            srCodesysWriteVars.SendValues();
        }

        catch
        {
            HasError = true;
        }
    }

    public void OnString_IPField(string newIP)
    {
        ConnectionIP = newIP;
        Debug.Log("New IP set!");
    }

    IEnumerator ShowError() //show error if connection hangs
    {
        GUI.Label(new Rect(Screen.width / 2, Screen.height / 2, 200f, 200f), "Coul
dn't send data, check connection settings");
        yield return new WaitForSeconds(5.0f);
        ErrorShown = false;
    }
}
```

LIITE VI: CarManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using System;

public class CarManager : MonoBehaviour
{

    //objects for spawning and handling the cars
    public GameObject Car_with_AI;
    private GameObject thisCar;
    public GameObject SpawnThing;

    public class ParkPlace //class for parking places
    {
        public GameObject place;
        public bool occupied;
        public int occupiedBy;
    }

    public List<ParkPlace> ParkPlaces1 = new List<ParkPlace>(); //List of parkplac
e objects 1st floor
    public List<ParkPlace> ParkPlaces2 = new List<ParkPlace>(); //List of parkplac
e objects 2nd floor
    public List<ParkPlace> ParkPlaces3 = new List<ParkPlace>(); //List of parkplac
e objects 3rd floor

    public List<ParkPlace> EmptyParks1 = new List<ParkPlace>(); //List of empty pa
rkplaces 1st floor
    public List<ParkPlace> EmptyParks2 = new List<ParkPlace>(); //List of empty pa
rkplaces 2nd floor
    public List<ParkPlace> EmptyParks3 = new List<ParkPlace>(); //List of empty pa
rkplaces 3rd floor


    public List<GameObject> Carlist = new List<GameObject>(); //List of cars on th
e environment

    //public ParkPlace NewPark = new ParkPlace(); //temporary parkplace object for
 adding to the list

    private Transform parkTrans;

    public GameObject inObj;
    public GameObject outObj;
    public GameObject awayObj;
    public GameObject firstTransObj;

    public Vector3 spawnPoint = new Vector3(0, 0, 0);//Where the cars are spawned
    public Quaternion spawnRotation = new Quaternion(0,0,0,0); //Rotation of the s
pawning car

    private int maxCars = 120; //maximum amount of cars on the scene at once
    private float delayTime = 15.0f; //time to wait before spawning new car
    private int i = 0; //Car id counter
    private int avoidPriority = 0; //car navmesh avoiding priority for smooth queu
ing
```

```
    void Start()
    {
        GameObject[] ParkThings1 = GameObject.FindGameObjectsWithTag("PPlace"); //
Get all parking places (tagged)
        GameObject[] ParkThings2 = GameObject.FindGameObjectsWithTag("PPlace2");
        GameObject[] ParkThings3 = GameObject.FindGameObjectsWithTag("PPlace3");
        //Debug.Log("there is " + ParkThings.Length + " items on parkthings list")
;


        foreach (GameObject thisPark in ParkThings1) //add the places to a list
         {
            ParkPlaces1.Add(new ParkPlace() { place = thisPark, occupied = false,
occupiedBy = 0 });
         }

        foreach (GameObject thisPark in ParkThings2) //add the places to a list
        {
            ParkPlaces2.Add(new ParkPlace() { place = thisPark, occupied = false,
occupiedBy = 0 });
        }

        foreach (GameObject thisPark in ParkThings3) //add the places to a list
        {
            ParkPlaces3.Add(new ParkPlace() { place = thisPark, occupied = false,
occupiedBy = 0 });
        }
        spawnPoint = SpawnThing.transform.position;
        spawnRotation = SpawnThing.transform.rotation;
        InvokeRepeating("CreateCar", 1f, delayTime);
    }

    public Transform GiveDestination(int carNum, int floornum, int destType)
    {
        int getRand;
        ParkPlace park;

        switch (floornum)
        {
            case 1:
                EmptyParks1 = ParkPlaces1.FindAll(x => x.occupied == false); //fin
d all empty parking places on floor
                getRand = UnityEngine.Random.Range(1, EmptyParks1.Count); //choose
 random placce from floor
                park = EmptyParks1[getRand];
                break;

            case 2:
                EmptyParks2 = ParkPlaces2.FindAll(x => x.occupied == false);
                getRand = UnityEngine.Random.Range(1, EmptyParks2.Count);
                park = EmptyParks2[getRand];
                break;

            case 3:
                EmptyParks3 = ParkPlaces3.FindAll(x => x.occupied == false);
                getRand = UnityEngine.Random.Range(1, EmptyParks3.Count);
```

```
                park = EmptyParks3[getRand];
                break;

            default:
                park = null;
                break;
                //Debug.Log("There is " + EmptyParks.Count + " unoccupied items on
 parkplace list");
        }

        if (destType == 2 && park != null) // give way to an empty park
        {
            //Debug.Log(park.occupied);
            parkTrans = park.place.transform;
            park.occupied = true;
            park.occupiedBy = carNum;
            return parkTrans;
        }

        else if (destType == 1) //destination to in gate
        {
            return inObj.transform;
        }

        else if (destType == 3) //destination to out gate
        {
            return outObj.transform;
        }

        else //give destination to the out place
        {
            return awayObj.transform;
        }
    }

    void CreateCar() //spawns a new car prefab when called
    {
        if (Carlist.Count < maxCars && !this.GetComponentInChildren<IOScript>().Ha
llIsFull)
        {
            i = i+1;
            if (avoidPriority < 99)
            {
                avoidPriority++;
            }

            else
            {
                avoidPriority = 1;
            }

            thisCar = Instantiate(Car_with_AI, spawnPoint, spawnRotation) as GameO
bject;
            thisCar.GetComponent<AutoAI>().carNumber = i;
            thisCar.GetComponent<AutoAI>().newDestination = firstTransObj.transfor
m;
            thisCar.GetComponent<NavMeshAgent>().avoidancePriority = avoidPriority
;
```

```
            Carlist.Add(thisCar);
        }
    }
}
```

LIITE VII: RestartS.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class RestartS : MonoBehaviour
{
    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        Man_Call = IO_Thing.GetComponent<CarManager>();
        IO_Call = IO_Thing.GetComponent<IOScript>();
    }
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.R))
        {
            RestartGame();
        }

        if (Input.GetKeyDown(KeyCode.P))
        {
            TogglePause();
        }

    }
    public void RestartGame() // resets all the cars and pauses game
    {
        foreach (GameObject x in Man_Call.Carlist)
        {
            Destroy(x.gameObject);
        }

        Man_Call.Carlist.Clear();

        Time.timeScale = 0f;
        IO_Call.ToggleNetVars();
    }
    public void TogglePause() //used to toggle the simulation run time
    {
        if (Time.timeScale == 1f)
        {
            Time.timeScale = 0f;
        }

        else
        {
            Time.timeScale = 1f;
        }

        IO_Call.ToggleNetVars(); //toggle network variable updates on/off when unp
aused/paused
    }
}
```

LIITE VIII: CarOnGate.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarOnGate : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    public AutoAI thisCar;
    private int getRand;
    private bool bPressed;

    void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        Man_Call = IO_Thing.GetComponent<CarManager>();
    }

    void OnTriggerEnter (Collider other)
    {
        if (other.tag == "Car") //check for car only
        {
            thisCar = other.gameObject.GetComponent<AutoAI>();
            thisCar.carAgent.isStopped = true; // stop the car

            IO_Call.CarOnGateIn = true; //change IO value

            if (IO_Call.HallIsFull)
            {
                thisCar.newDestination = Man_Call.GiveDestination(thisCar.carNumbe
r, getRand, 0); //if hall is full go straight to the end
                thisCar.GetComponentInChildren<CarBumper>().queuing = 0;
            }

            else
            {
                StartCoroutine(GetButtonPress(2f, 1f)); //press the button for tic
ket
            }

        }
    }

    void OnTriggerStay(Collider other)
    {
        if (other.tag == "Car" && bPressed && IO_Call.GInIsOpen && !IO_Call.CarLet
In) //drive in when gate is open
        {

            {
                getRand = UnityEngine.Random.Range(1, IO_Call.floorsactive); //get
 a floor from active ones
                thisCar.newDestination = Man_Call.GiveDestination(thisCar.carNumbe
r,getRand,2); //get a parking place
                thisCar.floorNumber = getRand;
```

```
                thisCar.GetComponentInChildren<CarBumper>().queuing = 0;
                Invoke("WakeCar", 2f);
                IO_Call.CarLetIn = true;
                bPressed = false;
            }
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Car")
        {
            IO_Call.CarOnGateIn = false;
            IO_Call.CarLetIn = false;
        }
    }

    IEnumerator GetButtonPress(float ttillpress, float tofpress)
    {
        yield return new WaitForSeconds(ttillpress);
        IO_Call.GetTicket = true;
        bPressed = true;
        yield return new WaitForSeconds(tofpress);
        IO_Call.GetTicket = false;
    }

    void WakeCar() //starts the car movement
    {
        thisCar.carAgent.isStopped = false;
    }
}
```

LIITE IX: CarOnGateOut.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarOnGateOut : MonoBehaviour
{

    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    public AutoAI thisCar;
    private bool bPressed;

    void Start()
    {
        //locate the IO script and manager for handling
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        Man_Call = IO_Thing.GetComponent<CarManager>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Car")
        {
            IO_Call.CarOnGateOut = true;
            StartCoroutine(GetButtonPress(2f, 1f)); //press the button for ticket
        }
    }

    void OnTriggerStay(Collider other)
    {
        if (other.tag == "Car" && bPressed && IO_Call.GOutIsOpen && !IO_Call.CarLe
tOut)
        {
            thisCar = other.gameObject.GetComponent<AutoAI>();

            thisCar.newDestination = Man_Call.GiveDestination(thisCar.carNumber, 0
,0);
            Invoke("WakeCar", 2f);
            IO_Call.CarLetOut = true;
            bPressed = false;
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Car")
        {
            IO_Call.CarOnGateOut = false;
            IO_Call.CarLetOut = false;
        }
    }

    IEnumerator GetButtonPress(float ttillpress, float tofpress)
    {
        yield return new WaitForSeconds(ttillpress);
```

```
        IO_Call.PaymentComplete = true;
        bPressed = true;
        yield return new WaitForSeconds(tofpress);
        IO_Call.PaymentComplete = false;
    }
}
```

LIITE X: GateMotorIn.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GateMotorIn : MonoBehaviour
{
    public GameObject IO_Thing;
    public IOScript IO_Call;
    private HingeJoint hinge;
    private int mForce = 10;
    private int mSpeed = 10;

    public void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        hinge = this.GetComponent<HingeJoint>();
        hinge.useMotor = true;

        JointMotor hmotor = hinge.motor;
        hmotor.force = mForce;
        hmotor.freeSpin = false;
        hmotor.targetVelocity = 0;
        hinge.motor = hmotor;
    }

    public void FixedUpdate()
    {
        if (IO_Call.CloseGIn)
        {
            CloseGate(hinge);
        }

        else if (IO_Call.OpenGIn)
        {
            OpenGate(hinge);
        }

        else
        {
            StopGate(hinge);
        }
    }

    void OpenGate(HingeJoint gHinge)
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = -mSpeed;
        gHinge.motor = gMotor;
    }

    void CloseGate(HingeJoint gHinge)
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = mSpeed;
        gHinge.motor = gMotor;
    }
    void StopGate(HingeJoint gHinge)
```

```
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = 0;
        gHinge.motor = gMotor;
    }
}
```

LIITE XI: GateMotorOut.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GateMotorout : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;
    private HingeJoint hinge;
    private int mForce = 10;
    private int mSpeed = 10;

    public void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        hinge = this.GetComponent<HingeJoint>();
        hinge.useMotor = true;

        JointMotor hmotor = hinge.motor;
        hmotor.force = mForce;
        hmotor.freeSpin = false;
        hmotor.targetVelocity = 0;
        hinge.motor = hmotor;
    }

    public void FixedUpdate()
    {
        if (IO_Call.CloseGOut)
        {
            CloseGate(hinge);
        }

        else if (IO_Call.OpenGOut)
        {
            OpenGate(hinge);
        }

        else
        {
            GateStop(hinge);
        }
    }

    void OpenGate(HingeJoint gHinge)
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = -mSpeed;
        gHinge.motor = gMotor;
    }

    void CloseGate(HingeJoint gHinge)
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = mSpeed;
        gHinge.motor = gMotor;
    }
```

```
    void GateStop(HingeJoint gHinge)
    {
        JointMotor gMotor = gHinge.motor;
        gMotor.targetVelocity = 0;
        gHinge.motor = gMotor;
    }
}
```

LIITE XII: GInOpenTS.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GInOpenTS : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
    }

    void OnTriggerEnter(Collider other)
    {
        IO_Call.GInIsOpen = true;
    }

    private void OnTriggerExit(Collider other)
    {
        IO_Call.GInIsOpen = false;
    }

}
```

LIITE XIII: GOutOpenTS.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GOutOpenTS : MonoBehaviour
{

    public GameObject IO_Thing;
    public IOScript IO_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
    }

    void OnTriggerEnter(Collider other)
    {
        IO_Call.GOutIsOpen = true;
    }

    private void OnTriggerExit(Collider other)
    {
        IO_Call.GOutIsOpen = false;
    }
}
```

LIITE XIV: GInCloseTS.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GInCloseTS : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("GateBar")) //only activate for the gate b
oom (tagged)
            IO_Call.GInIsClosed = true;
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject.CompareTag("GateBar"))
            IO_Call.GInIsClosed = false;
    }
}
```

LIITE XV: GOutCloseTS.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GOutCloseTS : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("GateBar")) //only activate for the gate b
oom (tagged)
            IO_Call.GOutIsClosed = true;
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject.CompareTag("GateBar"))
            IO_Call.GOutIsClosed = false;
    }
}
```

LIITE XVI: CarBumper.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class CarBumper : MonoBehaviour {

        public NavMeshAgent parentAgent;
        private float wTime;
    public int queuing = 1;

        void Start ()
        {
                parentAgent = this.GetComponentInParent<NavMeshAgent>();
        }

        private void OnTriggerEnter(Collider other)
        {
                if (other.tag == "Car")
                {
                        StopCar();
                }

        }

        IEnumerator OnTriggerStay(Collider other)
        {
                if (other.tag == "Car" && queuing == 0)
                {
                        wTime = Random.Range(5.0f, 10.0f);

        Invoke("DriveForward", wTime);//if cars are stuck towards one another, sta
rt driving anyways if no movement has happened
                        Invoke("StopCar", 0.3f);
                }
                yield return new WaitForSeconds(wTime*1.5f);
        }

        private void OnTriggerExit(Collider other)
        {
                if (other.tag == "Car")
                {
        CancelInvoke("DriveForward"); //cancel the invoke for stuck cars before ca
lling new invoke
                        Invoke("DriveForward", 1.5f);
                }
        }

        private void DriveForward()
        {
                parentAgent.isStopped = false;
        }

        private void StopCar()
        {
                parentAgent.isStopped = true;
        }
}
```

LIITE XVII: CarParked.cs

```
public class CarParked : MonoBehaviour
{
    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    public int parkFloor;
    private float ParkTimer;
    private AutoAI thisCar;
    private CarManager.ParkPlace thisListed;

    void Start()
    {
        //locate the IO script and manager script for handling
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        Man_Call = IO_Thing.GetComponent<CarManager>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Car")
        {
            ParkTimer = UnityEngine.Random.Range(18f, 36f); //random parking time
            thisCar = other.gameObject.GetComponent<AutoAI>();
            StartCoroutine(CarParkfunc());
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Car")
        {
            thisCar = other.gameObject.GetComponent<AutoAI>();
            parkFloor = thisCar.floorNumber;
            switch (parkFloor) // remove park occupation tag for new allocation
            {
                case 1:
                    thisListed = Man_Call.ParkPlaces1.Find(x => x.occupiedBy == th
isCar.carNumber);
                    break;
                case 2:
                    thisListed = Man_Call.ParkPlaces2.Find(x => x.occupiedBy == th
isCar.carNumber);
                    break;
                case 3:
                    thisListed = Man_Call.ParkPlaces3.Find(x => x.occupiedBy == th
isCar.carNumber);
                    break;
                default:
                    break;
            }

            thisListed.occupied = false;
        }
    }
        IEnumerator CarParkfunc() //wait for the defined (random) time and drive t
o the out gate
```

```
   {
        yield return new WaitForSeconds(ParkTimer);
        thisCar.newDestination = Man_Call.GiveDestination(thisCar.carNumber, 0, 3)
;
    }
}
```

LIITE XVIII: Indicator_light.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Indicator_light : MonoBehaviour {

    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    public Renderer IsFullI;
    public Material IsFullOn;
    public Material IsFullOff;

    public Renderer HasRoomI;
    public Material HasRoomOn;
    public Material HasRoomOff;

    void Start ()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        Man_Call = IO_Thing.GetComponent<CarManager>();

        IsFullI = this.transform.Find("Red_Dome").GetComponent<Renderer>();
        HasRoomI = this.transform.Find("Green_Dome").GetComponent<Renderer>();
    }


        void Update ()
    {
        if (IO_Call.HallHasRoom)
        {
            HasRoomI.material = HasRoomOn;
        }

        else
        {
            HasRoomI.material = HasRoomOff;
        }

        if (IO_Call.HallIsFull)
        {
            IsFullI.material = IsFullOn;
        }

        else
        {
            IsFullI.material = IsFullOff;
        }

    }
}
```

LIITE XIX: DestroyCar.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DestroyCar : MonoBehaviour
{
    public GameObject IO_Thing;
    public IOScript IO_Call;
    public CarManager Man_Call;

    private void Start()
    {
        IO_Thing = GameObject.Find("IO_Object");
        IO_Call = IO_Thing.GetComponent<IOScript>();
        Man_Call = IO_Thing.GetComponent<CarManager>();
    }
    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Car")
        {
            if (other.gameObject.GetComponent<AutoAI>().carNumber != 0)
            {
                IO_Call.CarsThrough = IO_Call.CarsThrough + 1;
            }

            Man_Call.Carlist.Remove(other.gameObject);
            Destroy(other.gameObject);
        }

    }


}
```

LIITE XX: GVLfile.GVL

```
<GVL>
  <Declarations><![CDATA[VAR_GLOBAL
                CarOnGateIn: BOOL;
                GateInOpen: BOOL;
                GateInClosed: BOOL;
                CarOnGateOut: BOOL;
                GateOutOpen: BOOL;
                GateOutClosed: BOOL;
                CarLetIn: BOOL;
                CarLetOut: BOOL;
END_VAR]]></Declarations>
  <NetvarSettings Protocol="UDP">
    <ListIdentifier>1</ListIdentifier>
    <Pack>TRUE</Pack>
    <Checksum>FALSE</Checksum>
    <Acknowledge>FALSE</Acknowledge>
    <CyclicTransmission>TRUE</CyclicTransmission>
    <TransmissionOnChange>FALSE</TransmissionOnChange>
    <TransmissionOnEvent>FALSE</TransmissionOnEvent>
    <Interval>T#50ms</Interval>
    <MinGap>T#20ms</MinGap>
  </NetvarSettings>
</GVL>
```

LIITE XXI: NVLSend

```
//This gobal variable list is sent via the network.
//Protocol: UDP
//The variables are in a defined order, don't change! Names can be changed if desired.

VAR_GLOBAL
   OpenGIn: INT;
            CloseGIn: BOOL;
            OpenGOut: BOOL;
            CloseGOut: BOOL;
            HallIsFull: BOOL;
            HallHasRoom: BOOL;
END_VAR
```