Jussi Lukkarinen

Virtuaalipelaajan tekoälyn päättelyarkkitehtuuri vuoropohjaisessa strategiapelissä

Opinnäytetyö
Kajaanin ammattikorkeakoulu
Tradenomi
Tietojenkäsittelyn koulutusohjelma

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Luonnontieteet	Koulutusohjelma Tietojenkäsittely
Tekijä(t) Jussi Lukkarinen	
Työn nimi Virtuaalipelaajan tekoälyn päättelyarkkitehtuuri vuoropohjaisessa strategiapelissä	
Vaihtoehtoiset ammattiopinnot	Ohjaaja(t) Mikko Romppainen
	Toimeksiantaja
Aika 09.04.2014	Sivumäärä ja liitteet 35 + 2

Opinnäytetyön tavoitteena oli suunnitella ja kehittää virtuaalipelaajan tekoälyn päättelyarkkitehtuuri vuoropohjaiseen strategiapeliin. Opinnäytetyön aikana toteutettiin koko tekoäly, mutta opinnäytetyö päätettiin rajata päättelyarkkitehtuuriin ja tekoälyn päätöksentekoon liittyviin osa-alueisiin. Tekoälyä tehtiin Planet Cube -videopeliä varten, joka on tarkoitus julkaista myöhemmin vielä määrittelemättömänä ajankohtana.

Opinnäytetyön alussa esitellään erilaisia tekoälyn tyyppejä ja niiden välisiä eroja. Tämän jälkeen opinnäytetyössä käsitellään tekoälyissä käytettyjä tekniikoita, joita on hyödynnetty tekoälyn toteutuksessa. Opinnäytetyön kannalta merkittävin on ADAPTA-päättelyarkkitehtuuri (Allocation and Decomposition Architecture for Performing Tactical AI), jonka pohjalta opinnäytetyön aikana kehitettyä tekoälyä on lähdetty toteuttamaan.

Tekoälytekniikoiden jälkeen esitellään Planet Cube -peliprojekti, jotta voidaan määritellä ympäristö, jossa tekoälyn tulee toimia. Peliprojektin esittelyssä selitetään pelin konseptin lisäksi pelin eteneminen sekä pelimaailmassa vaikuttavat säännöt. Näiden pohjalta pystytään paremmin määrittelemään toiminnot, joita tekoälyn tulee suorittaa ollakseen haastava vastustaja ihmispelaajalle, ja suunnittelemaan tekoälyn eri osien toiminta.

Tekoälyn toteutussuunnitelman jälkeen esitellään, minkälaisiin teknisiin ratkaisuihin tekoälyn toteutuksessa päädyttiin ja miksi nämä ratkaisut tehtiin. Tämän jälkeen esitellään kuinka tekoälyn toimintaa testattiin ja minkälaisiin tuloksiin testauksen aikana päädyttiin. Lopuksi opinnäytetyössä käsitellään ne tekoälyn toteutuksen osat, jotka aiheuttivat ongelmia tekoälyn toiminnassa ja joihin tulisi jatkokehityksen aikana kiinnittää huomioita.

Kieli	Suomi
Asiasanat	Tekoäly, päättelyarkkitehtuuri, ohjelmointi
Säilytyspaikka	☒ Verkkokirjasto Theseus ☐ Kajaanin ammattikorkeakoulun kirjasto

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THESIS
ABSTRACT

School Business	Degree Programme Business Administration
Author(s) Jussi Lukkarinen	
Title Reasoning Architecture of Virtual Player Artificial Intelligence for Turn-based Strategy Game	
Optional Professional Studies	Instructor(s) Mikko Romppainen
	Commissioned by
Date 09.04.2014	Total Number of Pages and Appendices 35 + 2

The goal of this thesis was to design and implement a reasoning architecture of virtual player artificial intelligence for a turn-based strategy game. During the thesis, the entire artificial intelligence was implemented but the focus of the thesis was limited to reasoning architecture and other decision making related fields. The artificial intelligence was developed for the Planet Cube -video game. The release date of the game has not been decided yet.

The thesis begins with an overview of different types of artificial intelligence and the differences between them. After that, different kinds of techniques utilized in artificial intelligence development are reviewed. The techniques are limited to those utilized in the artficial intelligence developed during the thesis. From the point of view of this thesis, the most significant of these techniques is the ADAPTA-reasoning architecture (Allocation and Decom-position Architecture for Performing Tactical AI). The reasoning achitecture and artificial intelligence developed during the thesis are based on ADAPTA.

After reviewing the techniques, the Planet Cube -game project will be introduced. This introduction defines the environment in which the artificial intelligence has to perform in. The introduction will describe the concept of the game, the flow of the gameplay and the rules that apply in the game world. Based on the introduction, it is easier to define and design the functionalities of the artificial intelligence required to provide a challenging opponent for a human player.

The thesis ends with a review of the technical solutions used in the implementation of the artificial intelligence and explanations to why these solutions were selected. Also, the process of testing the artificial intelligence and the results of the testing are described. As a conclusion, problems encountered during the testing and possible solutions to these problems are reviewed.

Language of Thesis	Finnish
Keywords	Artificial intelligence, reasoning architecture, programming
Deposited at	☒ Electronic library Theseus ☐ Library of Kajaani University of Applied Sciences

SISÄLLYS

SYMBOLILUETTELO

Akateeminen tekoäly	Älykkyyttä vaativia ongelmia ratkova tekoäly.
Hakutaulukko	Tietorakenne, joka yhdistää avainarvot todellisiin arvoihin.
Moduuli	Ohjelmoinnissa moduuli tarkoittaa tietokoneohjelman itsenäistä osaa, jolla on oma toiminnallinen tehtävä.
Navigaatioverkko	Tekoälyissä käytetty tietorakenne, joka helpottaa reitinhakua suurissa tiloissa.
Solmugraafi	Solmuista koostuva verkko. Solmugraafia voidaan käyttää apuna tekoälyjen reitinhaussa.
Strategia	Pitkän aikavälin suunnitelma jonkin tavoitteen saavuttamiseksi.
Taktiikka	Lyhyen aikavälin toimintatapa jonkin tavoitteen saavuttamiseksi.
Tekoälyagentti	Tekoälyn ohjaama olento tai entiteetti.

1 JOHDANTO

Tekoäly tuo lisäarvoa videopeleihin tarjoamalla pelaajalle haastavia ja älykkäitä vastustajia. Strategiapeleissä tekoälyn ohjaama virtuaalipelaaja pelaa ihmispelaajaa vastaan toisen pelaajan paikalla silloinkin, kun toista ihmispelaajaa ei ole saatavilla. Strategiapeleissä käytettävien laadukkaiden tekoälyjen kehittäminen on haastavaa ja aikaa vievää, sillä peleissä olevat strategiset ongelmat ovat usein hyvin monimutkaisia. Tässä opinnäytetyössä suunnitellaan ja kehitetään tekoälyn päättelyarkkitehtuuri vuoropohjaiseen strategiapeliin. Prosessin aikana kehitettiin kokonainen tekoäly, mutta kirjallinen osuus on rajattu päättelyarkkitehtuuriin ja muihin päätöksen tekoon liittyviin tekijöihin.

Opinnäytetyössä käsitellään strategiapeleissä käytettyjen virtuaalipelaajien kehittämisessä hyödynnettäviä menetelmiä sekä niiden hyviä ja huonoja puolia. Myöhemmin tässä opinnäytetyössä näitä menetelmiä hyödynnetään strategiapelin tekoälyn suunnittelussa ja toteutuksessa. Suunnitteluvaiheessa otetaan huomioon pelin luomat tarpeet, jotka tekoälyn tulisi tyydyttää. Tavoitteena on toteuttaa tekoäly, joka tarjoaisi haastavan vastustajan ihmispelaajalle ilman, että tekoäly turvautuu pelissä huijaamiseen.

Toteutettu tekoäly kehitettiin Planet Cube -strategiapeliin, joka on kehitetty Unity 3D-pelimoottorilla PC-ympäristöön. Planet Cube -videopeli oli vielä opinnäytetyön kehityksen aikana keskeneräinen, joten tämä näkyy myös kehitetyssä tekoälyssä. Suunnittelussa ja toteutuksessa on pyritty ottamaan huomioon pelin kehityksen vaihe. Tekoälyn tulee olla helposti laajennettavissa vastaamaan pelin tarpeita nyt ja tulevaisuudessa.

2 TEKOÄLY PELEISSÄ

Peleissä käytettyjen tekoälyjen kehityksessä pääpaino on usein tekoälyn käyttäytymisessä kognitiivisen toiminnan sijaan. Peleissä on usein tärkeämpää, että tekoäly vaikuttaa tekevän järkeviä päätöksiä kuin se, että tekoälyn kognitiivinen toiminta olisi erityisen kehittynyttä. Peleissä käytetyissä tekoälyissä oleellisempaa onkin tekoälyn käyttäytyminen kuin ne vaiheet, jotka johtivat saavutettuun käyttäytymiseen. Tämä korostuu erityisesti tarkastellessa tekoälyn toimintaa kuluttajan näkökulmasta. Kuluttaja on harvoin kiinnostunut tekoälyn toteutuksessa käytetyistä algoritmeista ja menetelmistä. Sen sijaan kuluttaja on kiinnostunut tekoälystä, joka tarjoaa älykkäältä vaikuttavan haastajan tai kumppanin, joka tuo viihdyttävää lisäarvoa pelille, riippumatta siitä, millä tavalla tekoälyn näennäinen älykkyys on saavutettu. (Schwab B 2004, 4-7.)

Kun tavoitteena on kehittää näennäisesti älykkäästi toimiva tekoäly, voi kehittäjä turvautua ratkaisuihin, joissa tekoälyllä on mahdollisuus rikkoa pelissä vallitsevia sääntöjä, joita pelaajan täytyy noudattaa omaa toimintaansa suunnitellessaan. Tällaiset ratkaisut olivat hyvin yleisiä varhaisemmissa tietokonepeleissä, sillä sen aikakauden tietokoneet olivat huomattavasti nykyaikaisia tietokoneita rajoittuneempia laskentatehon ja muistin määrässä. Näin voitiin usein vähentää tekoälyn tietokoneelle aiheuttamaa kuormitusta, jolloin resursseja jäi enemmän muiden pelin toimintojen käyttöön. Ammuntapelin tekoälyagentti voisi esimerkiksi olla koko ajan tietoinen pelaajan sijainnista ja siten välttyä pelaajan etsimiseltä. Tämä säästäisi jonkin verran tietokoneen tehoja, mutta tekoälyagenttia vastaan pelaava pelaaja alkaisi todennäköisesti epäillä jonkin olevan vialla tekoälyagentin toimien perusteella. (Schwab B 2004, 4-7.)

Pelin sääntöjä rikkova tekoäly on kuitenkin nykypäivänä harvinaistuva toteutustapa, sillä tällaisia ratkaisuja hyödyntävät tekoälyt koetaan usein vähemmän miellyttäviksi, koska pelaaja usein kokee tulleensa huijatuksi. Lisäksi nykyaikaisten tietokoneiden kasvanut laskentateho ja muistin määrä ovat mahdollistaneet kehittyneempien tekniikoiden käyttöönoton pelien tekoälyjen kehityksessä. Peleissä käytetyt tekoälyt alkavatkin lähestyä toiminnassaan ja käyttämissään tekniikoissa akateemisia tekoälyjä. (Schwab B 2004, 4-7.)

Virtuaalipelaaja

Virtuaalipelaaja on tekoälyagentti, jonka on tarkoitus toimia ihmispelaajan tavoin ihmispelaajan tilalla. Pelaaja voi havaita virtuaalipelaajan olemassa olon ainoastaan virtuaalipelaajan te-

kemien päätöksien ja toimintojen kautta, sillä virtuaalipelaaja harvoin esiintyy pelimaailmassa visuaalisesti. Virtuaalipelaaja pyrkii toimimaan ihmispelaajan tavoin ja tarjoamaan miellyttävän vastustajan moninpeleissä, jos toista ihmispelaajaa ei ole saatavilla. (Ahlquist J & Novak J 2008, 12-17.)

Virtuaalipelaajia on kehitetty lautapeleihin sekä digitaalisiin strategiapeleihin. Strategiapelit ovat hyödyntäneet virtuaalipelaajia jo pitkän aikaa tarjoamaan pelaajalle haastajan ilman toisen ihmispelaajan läsnäoloa. Toisin kuin reaaliaikaisissa strategiapeleissä, vuoropohjaisissa strategiapeleissä yksittäisillä yksiköillä on hyvin vähän tai ei lainkaan älykkyyttä ja tekoälyn toiminta on puhtaasti virtuaalipelaajan vastuulla. (Ahlquist J & Novak J 2008, 12-17.)

3 STRATEGIAPELEISSÄ YLEISESTI KÄYTETTYJÄ TEKNIIKOITA

Vuoropohjaisen strategiapelin tietokonepelaajan tekoälyssä korostuvat jotkin niistä tutkimuksellisista haasteista, jotka on esitelty artikkelissa RTS Games as Test-Bed for Real-Time AI Research (Buro M & Furtak T 2003). Näitä haasteita tarkastelemalla saa hyvän kuvan muuttujista ja tekijöistä, jotka vaikuttavat tekoälyn tekemiin päätöksiin ja siten antavat myös lähtökohdan tekoälyssä käytettävien tekniikoiden valinnalle. Buron ja Furtakin määrittelemistä tutkimushaasteista huomiota vaativat seuraavat:

- päätöksenteko epävarmoissa tilanteissa
- spatiaalinen ja temporaalinen päättely
- resurssien hallinta
- yhteistyö
- mukautuvuus. (Bergsman M & Spronck P 2008.)

Maurice Bergsman ja Pieter Spronckin vuonna 2008 esittelemä ADAPTA-päättelyarkkitehtuuri yhdistettynä muihin tekoälyissä käytettyihin tekniikkoihin, kuten vaikutuskarttoihin, antaa hyvän pohjan näihin haasteisiin vastaamiselle. (Bergsman M & Spronck P 2008.)

3.1 Strategiapelin tekoälyn tavoitteita ja tarpeita

Strategiapelien tekoälyissä käytetään lisääntyvissä määrin tekoälytekniikoita, joiden tarkoituksena on tehdä tekoälyn toiminnasta aidosti älykkäämpää, eikä ainoastaan antaa pelaajalle vaikutelmaa älykkyydestä (Schwab B 2004, 4-7). Tietoa keräämällä ja analysoimalla mahdollistetaan tekoälyn toiminnan pohjautuminen pelissä vallitsevan tilanteen tulkintaan ennalta määrättyjen toimintamallien sijasta (Schwab B 2004, 101-102).

Maastoanalyysillä tarkoitetaan tekoälytekniikkaa, jossa peliympäristö pelkistetään helpommin käsiteltävään muotoon ja tätä tietoa analysoimalla pyritään kehittämään tehokkaampia ratkaisuja. Maastoanalyysin pohjana ovat usein vaikutuskartat, joiden avulla tekoäly pystyy käsittelemään monimuotoista tietoa peliympäristöstä. (Schwab B 2004, 101.)

Vastustajan mallintaminen tarkoittaa tiedon keräämistä vihollisen joukkueesta ja vihollispelaajan käyttäytymismalleista. Kerätystä tiedosta saadaan parempi kokonaiskuva vastustajasta pelaajana ja vastustajan käytössä olevista resursseista. Vastustajan mallintamista voidaan käyttää hyödyksi peleissä, joissa puutteellinen tieto on osana strategista ongelmaa. Toinen käyttökohde tälle tekniikalle ovat pelit, joissa tekoälyn halutaan mukautuvan pelaajan toistamiin rutiineihin. Kerättyä tietoa käytetään vihollispelaajan toimien ennakoimiseen ja apuna tekoälyn omien taktiikoiden ja strategioiden luonnissa. Puutteellista tietoa sisältävissä peleissä on myös tärkeää, että tekoälyllä on keinoja tiedon keräämiseen. Tältä tosin voidaan välttyä, mikäli käytetään aikaisemmin käsiteltyä tapaa, jossa tekoäly rikkoo pelin sääntöjä. Kerättävää tietoa voivat olla esimerkiksi vihollisyksiköiden sijainnit ja vihollisjoukkueen kokoonpano. (Schwab B 2004, 101-102.)

Strategiapeleissä on usein jonkinlainen ekonomia ja resurssit, joiden hallinta on tärkeä osa tekoälyn toimintaa. Resurssit voidaan jakaa kahteen ryhmään, pääkäyttöiset raakaresurssit ja toissijaiset resurssit. Raakaresursseilla tarkoitetaan resursseja, jotka toimivat raaka-aineina toissijaisten resurssien rakentamiselle tai hankinnalle. Tällaisia resursseja voisivat olla esimerkiksi ruoka, puu, rauta ja muut vastaavat hyödykkeet. Strategiapeleissä toissijaisia resursseja voivat olla esimerkiksi tekoälyn hallinnassa olevat yksiköt ja rakennukset. Nimestään huolimatta toissijaiset resurssit ovat hyvin tärkeitä niitä hallitsevalle pelaajalle. Resursseja hallinnoitaessa on tärkeää, että rajallisia resursseja osataan allokoida oikeisiin käyttökohteisiin eikä niitä tuhlata sellaisiin kohteisiin, jotka eivät sillä hetkellä tai tulevaisuudessa edesauta tekoälyn strategisen tavoitteen saavuttamista. Resursseja hyvin hallitsevan tekoälyn täytyy pystyä mukautumaan muuttuvaan pelitilanteeseen. (Schwab B 2004, 102.)

Tässä osiossa esitetyt tarpeet korostavat korkean tason strategisten päätösten tärkeyttä. Matalan tason taktisiin päätöksiin keskittyvä tekoäly voi pärjätä hyvin yksittäisissä taisteluissa, mutta joutua ajan kuluessa alakynteen esimerkiksi huonon resurssien hallinnan ja allokoinnin vuoksi. On kuitenkin huomioitava, että mikään strategia ei takaa voittoa, jos strategian toteuttamiseen vaadittavissa tilanteissa ei suoriuduta. Tästä syystä taktisillakin päätöksillä on suuri vaikutus lopputulokseen. Kirjassaan AI Game Engine Programming Brian Schwab listaa seuraavat tekoälylle strategisesti tärkeät toiminnot:

a) Yksiköiden oikeanlainen ryhmittäminen ja oikeiden yksiköiden valinta taistelemaan vihollisen yksiköitä vastaan

b) Hyökkäys- ja puolustuslinjojen luominen

c) Maastonmuotojen hyödyntäminen taistelussa

d) Vetäytymisten suunnitteleminen ja oikea ajoittaminen

e) Väijytyksien hyödyntäminen. (Schwab B 2004, 110-111.)

Tästä listasta voi havaita, kuinka monimuotoinen strategiapeleissä käytettyjen tekoälyjen ratkaisema ongelma on. Kaikki tekoälyt eivät suorita listassa esiintyviä toimintoja, mutta loistavasti suoriutuvien tekoälyjen voi odottaa käsittelevän jokaista listassa esiintyvää toimintoa jollakin tasolla. (Schwab B 2004, 110-111.)

3.2 Tekoälyn päätöksenteko

Pelissä käytetty tekoäly joutuu pelin aikana ratkomaan useita ongelmia ja tekemään monia päätöksiä. Peleissä tekoälyn päätöksenteon apuna voidaan käyttää tekniikoita, jotka pohjautuvat muun muassa sattumaan, oletuksiin, laskentaan tai ennalta määriteltyihin ehtoihin ja tietoon. On peli- ja tilannekohtaista, mitä näistä tekniikoista tulisi käyttää päätöksenteossa. Näitä tekniikoita voidaan tarvittaessa yhdistellä parhaan mahdollisen lopputuloksen aikaansaamiseksi. (Wetzel B 2008, 443.)

3.2.1 Satunnainen päätöksenteko

Satunnaisessa päätöksenteossa päätös valitaan satunnaisesti useista tarjolla olevista vaihtoehdoista. Satunnaisessa valinnassa käytetään apuna pelimoottoriin toteutettua satunnaislukugeneraattoria, joka tuottaa näennäisesti satunnaisia lukuja niitä tarvittaessa. Tekoälylle tarjolla olevista vaihtoehdoista voidaan valita yksi numeroimalla vaihtoehdot kasvavalla numeroinnilla (1, 2, 3, …, n) ja pyytämällä satunnaislukugeneraattorilta yksi luku näiden lukujen väliltä. Satunnaislukugeneraattorin tuottamalla luvulla numeroitu vaihtoehto valitaan käytettäväksi. (Wetzel B 2008, 443-444.)

Satunnaisuuteen perustuvan päätöksenteon hyviä puolia ovat sen helppo suunnittelu ja toteutus. Lisäksi satunnaislukugeneraattorien tuottama laitteistokuormitus on yleisesti ottaen

vähäistä verrattuna monimutkaisempien tekniikoiden tuottamaan kuormitukseen. Satunnaisen päätöksenteon heikkoudeksi voidaan lukea sen käyttökohteiden rajallinen määrä. Mikäli tarkoituksena on kehittää älykkäältä vaikuttava tekoäly, satunnaista päätöksentekoa voidaan hyödyntää lähinnä tilanteissa, joissa on tarjolla useita vaihtoehtoja ja vaihtoehdoista jokainen on lähes yhtä hyvä kuin mikä tahansa toinen. (Wetzel B 2008, 443-444.)

Esimerkiksi satunnaiselle päätöksenteolle sopivasta käyttökohteesta voitaisiin esittää seuraavanlainen tilanne: Vuoropohjaisessa strategiapelissä tekoälyn hallitsema yksikkö on tilanteessa, jossa yksikkö on hyökkäysetäisyydellä kahdesta viholliskohteesta. Pelin säännöt estävät yksikköä hyökkäämästä molempien viholliskohteiden kimppuun yhden vuoron aikana. Yksikön hyökkäys aiheuttaa molempiin viholliskohteisiin lähes saman määrän vahinkoa. Tällaisessa tilanteessa ei ole merkitystä, kumman viholliskohteen kimppuun yksikkö vuoron aikana hyökkää, mutta jokin päätös on tehtävä. Tässä tilanteessa tekoäly voi valita sattumanvaraisesti viholliskohteen, jonka kimppuun yksikkö hyökkää, ja siitä huolimatta vaikuttaa älykkäältä.

3.2.2 Yksinkertaiset strategiat

Yksinkertaisilla strategioilla tarkoitetaan yksinkertaisiin sääntöihin perustuvia strategioita, jotka hyödyntävät laskentaa hyvin vähän ja siten eivät kuormittavat laitteistoa runsaasti. Useat pelit hyödyntävät yksinkertaisia strategioita sellaisten vihollisten tekoälyissä, joiden älykkyyden ja toiminnan ei oleteta olevan kehittynyttä ja suunnitelmallista. Esimerkiksi pelissä esiintyvä villi susi voisi noudattaa strategiaa ’hyökkää kaikkien, paitsi oman lauman jäsenten, kimppuun’ ja vaikuttaa käyttäytyvän luonnollisella tavalla. Aivan kuten satunnaisen päätöksenteon myös yksinkertaisten strategioiden hyötyjä ovat matala laitteistokuormitus sekä helppo suunnittelu. (Wetzel B 2008, 444-445.)

3.2.3 Laskentakaavat

Päätöksenteossa voidaan hyödyntää myös laskentakaavoja. Tässä yhteydessä laskentakaavalla tarkoitetaan jonkinlaista kaavaa, jolla voidaan määrittää kuinka kannattava harkittu ratkaisu on. Laskentakaavojen huonoja puolia ovat niiden haastava kehittäminen ja testaaminen. Lisäksi laskentakaavat ovat vaikeasti hallittavia, mikäli kaavaan vaikuttavien muuttujien luku-

määrä kasvaa suureksi. Tällaisissa tapauksissa voidaan laskentaa yksinkertaistaa muuttujien summausta hyödyntäen. Muuttujien summauksella tarkoitetaan laskentatapaa, jossa muuttujilla on jonkinlainen vaikutus loppusummaan ja nämä vaikutukset lasketaan yhteen. Laskentakaavoja voidaan käyttää tilanteissa, joissa käsiteltävä ongelma on esitettävissä numeerisesti. Strategiapeleissä laskentakaavoja hyödynnetään usein esimerkiksi yksiköiden voimasuhteita laskiessa. (Wetzel B 2008, 445-449.)

Esimerkkinä laskentakaavoista voidaan käyttää tilannetta, jossa kaksi yksikköä on aloittamassa taistelun toisiaan vastaan. Laskentakaavaa käyttämällä voitaisiin vertailla näiden yksiköiden voimasuhteita ja niiden avulla määritellä, onko taistelun aloittaminen kannattavaa ja kummalla on etulyöntiasema. (Wetzel B 2008, 445-449). Voimasuhteet voitaisiin määrittää laskemalla molempien yksiköiden voima esimerkiksi seuraavanlaisella kaavalla:

$$voima = elämäpisteet^2 + 2 * hyökkäysteho + puolustusteho$$

3.2.4 Hakutaulukot

Hakutaulukoihin säilötään pelin kehitysvaiheessa luotua tietoa, jota hyödynnetään päätöksen tekoon pelin ajoaikana. Taistelutilanteen lopputulosta ennustettaessa voitaisiin hakutaulukosta katsoa, kuinka tehokas hyökkäävä yksikkö on kohteena olevaa yksikköä vastaan. Hakutaulukoita voidaan käyttää tekoälyn toiminnan nopeuttamiseksi, sillä haku taulukosta on huomattavasti nopeampaa kuin arvojen laskeminen ajoaikana. Hakutaulukosta hakeminen tapahtuu usein vakioajassa käytetystä tietorakenteesta riippuen. Muita hakutaulukoiden hyötyjä ovat niiden helppo hallittavuus sekä yksinkertainen toteutus. Hakutaulukoissa on tosin vaikea hallita monimutkaisia muuttujakokonaisuuksia. (Wetzel B 2008, 449.)

3.2.5 Monte Carlo -metodi

Monte Carlo -metodi on empiirinen metodi, jossa arvioitava tilanne simuloidaan useita kertoja järjestelemässä ja simulaatioiden lopputulosjakaumasta saadaan selvitettyä todennäköisin lopputulos. Monte Carlo -metodilla on joitain vaatimuksia pelitoteutuksen suhteen, sillä tekoälyllä täytyy olla pääsy tarvittaviin järjestelmiin, joissa simulaatioita voidaan ajaa. Monte Carlo -metodi ei selvitä, onnistuuko jokin toteutettava asia, vaan sen sijaan sillä voidaan sel-

vittää tilastotietoihin pohjautuen, kuinka suurella todennäköisyydellä asia onnistuu tai epäonnistuu. Monte Carlo -metodia voidaan käyttää esimerkiksi taistelun arvioidun lopputuloksen selvittämiseksi. Taistelu toistetaan useita kertoja peräkkäin, ja saadusta tilastotiedosta voidaan selvittää voiton ja tappion todennäköisyydet. (Wetzel B 2008, 449-453.)

Tehovaatimuksiltaan Monte Carlo -metodi on huomattavasti edeltäviä menetelmiä vaativampi, sillä tarkempien tuloksien saamiseksi vaaditaan useita simulaatiokertoja. Reaaliaikaisissa käyttökohteissa tämä voisi olla ongelma, mutta ei vuoropohjaisissa strategiapeleissä. Nykyisten prosessorien suoritustehot ovat niin korkeat, että suurikin määrä skenaarioita voidaan simuloida suhteellisen lyhyessä ajassa. (Wetzel B 2008, 449-453.)

3.3 Spatiotemporaalinen päättely

Tärkeänä osana strategisen tekoälyn toiminnassa ja päätöksenteossa ovat paikan ja ajan hahmottaminen eli spatiotemporaalinen päättely. Usein peleissä käytetään spatiotemporaalisen päättelyn pohjana vaikutuskarttoja. Vaikutuskarttoja voidaan hyödyntää esimerkiksi joukkueiden hallitsemien alueiden taltioimiseen ja vaarallisten alueiden mallintamiseen. Tekoäly voi käyttää näitä tietoja oman toimintansa suunnitteluun. Esimerkiksi tekoäly voi hyödyntää tietoa vaarallisista alueista yksiköiden siirtämiseen turvallisemmille alueille. Vaikutuskarttoja voidaan käyttää myös mallintamaan ajan kulumista, jota voidaan hyödyntää ennusteiden tekemiseen ja menneen statistiikan tarkasteluun. (Champandard A 2011.)

3.3.1 Vaikutuskartat

Vaikutuskartat tarjoavat uuden näkökulman pelimaailmaan. Ne ovat usein monitasoisia tietorakenteita, joissa jokainen taso säilyttää erilaisia pelimaailman tilaan liittyviä tietoja. Vaikutuskartat yhdistävät säilömänsä tiedon sijainteihin pelimaailmassa. Vaikutuskarttojen toiminta-ajatus on, että pelimaailman tilaan vaikuttaville tapahtumille asetetaan vaikutuksen lähtöarvot, jotka sen jälkeen levitetään ympäröiviin alueisiin. Tämä tarkoittaa sitä, että pelimaailmaan tilaan vaikuttavan tapahtuman sattuessa vaikutuskarttaan lisätään tapahtumapisteestä ulospäin leviävä vaikutusarvo. Tällä tavoin erilaiset pelimaailman tapahtumat ovat havaitta-

vissa myös tapahtumapisteen ulkopuolella siitä leviävänä vaikutuksena. (Sweetser P 2004, 439-441.)

Vaikutuskartat voivat olla ruudukkoon pohjautuvia, tai ne voidaan sitoa pelissä jo käytettävään toisenlaiseen navigaatiotietoon, kuten solmugraafiin (node graph) tai navigaatioverkkoon (navigation mesh). Vaikutuskarttoja voidaan käyttää nykyisen tilanteen säilömisen lisäksi mennyttä aikaa kuvaavana statistiikkana, ja niiden avulla voidaan tehdä myös ennustuksia tulevaisuudesta. (Champandard A 2011.)

Vaikutuskarttoja käytetään strategiapeleissä usein joukkueiden hallitsemien alueiden määrittämiseen, ja tämän tiedon avulla pelikenttä voidaan jakaa esimerkiksi hyökkäyksen kannalta kiinnostaviin ja ei-kiinnostaviin alueisiin. Tällä tavoin tekoälyllä on mahdollisuus suunnitella yksiköidensä liikkeitä siten, että yksiköitä siirretään sellaisia alueita kohti, jotka ovat strategisesti tärkeitä. (Woodcock S 2002, 221-231.)

3.3.2 Maastoanalyysi

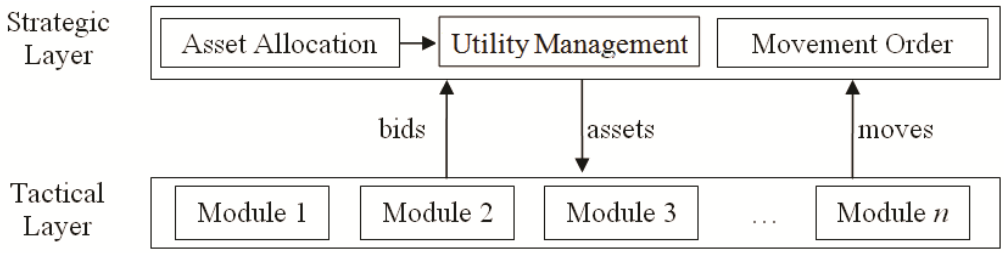
Maastoanalyysillä tarkoitetaan toimintoja, joilla tekoäly määrittää erilaisia maaston ominaisuuksia. Strategiapeleissä tekoäly on usein kiinnostunut maaston tarjoamasta suojasta sekä mahdollisista hyötyalueista, joista tekoälyn yksiköt saavat etulyöntiaseman vihollisia vastaan. Maastoa analysoidessa tekoäly pyrkii arvioimaan esimerkiksi vaikutuskarttoihin säilöttyä tietoa ja määrittämään sen pohjalta maaston laadullisia ominaisuuksia. Tällainen ominaisuus voisi olla esimerkiksi jonkin paikan tarjoaman suojan laatu. Maastoanalyysin avulla tekoälyagentti pystyy hyödyntämään maaston muotoja huomattavasti tehokkaammin kuin ilman sitä. (Schwab B 2004, 378-379.)

4 ADAPTA-PÄÄTTELYARKKITEHTUURI

ADAPTA (Allocation and Decomposition Architecture for Performing Tactical AI) on tekoälyn päättelyarkkitehtuuri, jossa tekoälyn toiminnot jaetaan pienempiin toimintoihin, jotta tekoäly olisi helpommin laajennettavissa ja se pysyisi kokonsa puolesta hallittuna. ADAPTA-päättelyarkkitehtuurissa tekoälyn toimintoja edustavat moduulit. Moduulit jaetaan kahteen tasoon riippuen siitä, minkälainen toiminto moduulin hoidettavana on. Nämä kaksi päättelyarkkitehtuurin tasoa ovat strateginen ja taktinen taso. (Bergsma M & Spronck P 2008.)

Taktisella tasolla toimivat moduulit eivät ole lainkaan kiinnostuneita strategisesta kokonaiskuvasta tai strategisista ongelmista ja keskittyvät ainoastaan oman tehtävänsä suorittamiseen. Koska tällaiset yksittäiset moduulit jakavat kuitenkin samat tekoälyn käytössä olevat resurssit ja yksittäisten moduulien toiminnan tulisi edesauttaa tekoälyn strategisia tavoitteita, ei toimintaa voida jättää ainoastaan taktisen tason moduulien hoidettavaksi. (Bergsma M & Spronck P 2008.)

Strategisella tasolla toimivien moduulien tehtävänä on pitää huolta korkeamman tason strategisesta toiminnasta. Strategisen tason moduulien vastuulla ovat pääasiassa resurssien hallinta ja taktisen tason moduulien toimintojen priorisointi. (Bergsma M & Spronck P 2008.)



Kuvio 1: Vuokaavio ADAPTA-päättelyarkkitehtuurista (Bergsma M & Spronck P 2008).

ADAPTA-päättelyarkkitehtuurin perustoimintamalli (Kuvio 1) on seuraavanlainen: aluksi strategisella tasolla toimiva resursseja allokoiva moduuli (Asset Allocator) tarjoaa käytettävissä olevia resursseja taktisella tasolla toimiville moduuleille. Taktisen tason moduulit luovat huutoja eri resursseista huutokaupan tapaan ja asettavat resursseille hyötyarvoja. Hyötyarvo kuvastaa sitä, kuinka tärkeänä taktisen tason moduuli resurssia pitää oman tavoitteensa saavuttamisen kannalta. Hyötyarvo määritetään moduulikohtaisesti, eikä moduulien luomien hyötyarvojen tarvitse olla suhteessa toisten moduulien laskemiin hyötyarvoihin. Taktisen ta-

son moduulit eivät ole kiinnostuneita muiden taktisen tason moduulien tavoitteista, ja moduulien luomat huudot tavoittelevat vain moduulin omaa etua. (Bergsma M & Spronck P 2008.)

Tämän jälkeen strategisella tasolla toimiva huutoja arvioiva (Utility Management) moduuli arvioi taktisen tason moduulien luomat huudot ja määrittää, millaisesta huutoyhdistelmästä olisi suurin yhteisöllinen hyöty. Moduuli siis pyrkii löytämään parhaan mahdollisen yhdistelmän huutoja siten, että huutojen yhteenlaskettu hyötyarvo olisi mahdollisimman suuri. Tämän jälkeen huutoja arvioiva moduuli jakaa huudetut resurssit taktisen tason moduulien käytettäväksi. (Bergsma M & Spronck P 2008.)

Seuraavaksi taktisen tason moduulit luovat siirrot käyttöönsä saamilleen resursseille. Varsinaiset siirrot luodaan tässä vaiheessa, jotta moduulit voisivat siirtoja luodessaan hyödyntää tietoa siitä, mitä resursseja moduulilla todellisuudessa on käytössään. Lopuksi taktisen tason moduulit luovuttavat luodut siirrot tekoälyn strategisella tasolla toimivalle siirtojen suoritusjärjestyksestä vastaavalle (Movement Order) moduulille. Siirtojen suoritusjärjestyksestä vastaavan moduulin tehtävänä on järjestellä luodut siirrot parhaaseen mahdolliseen suoritusjärjestykseen, ja tässä vaiheessa moduuli voi arvioida siirtojen suhteita toisiinsa. Siirtojen suoritusjärjestyksen määrittämisen jälkeen siirrot voidaan suorittaa ja tekoälyn vuoro on valmis. (Bergsma M & Spronck P 2008.)

Siirtojen suoritusjärjestyksestä vastaavan moduulin toiminnasta hyvä esimerkki olisi tilanne, jossa kaksi tekoälyn ohjaamaa yksikköä on hyökkäämässä yhtä vastustajan yksikköä vastaan. Ensimmäinen tekoälyn ohjaamista yksiköistä on tilanteessa, jossa yksikkö on vaarassa tuhoutua, mikäli vastustajan yksikkö vastaa hyökkäykseen omalla hyökkäyksellään. Mikäli toinen tekoälyn ohjaamista yksiköistä kykenee hyökkäämään vastustajan yksikön kimppuun ilman tuhoutumisen vaaraa, on kannattavaa arvioida, voitaisiinko vihollisen yksikkö tuhota molempien yksiköiden hyökkäyksillä. Mikäli tuhoaminen on mahdollista, on huomattavasti kannattavampaa asettaa ensimmäiseksi hyökkääjäksi yksikkö, joka ei ole tuhoutumisen vaarassa. Tällä tavoin ensimmäinen hyökkääjä voi vahingoittaa vastustajan yksikköä riittävästi, että tekoälyn ohjaamista yksiköistä jälkimmäisenä hyökkäävä yksikkö voi tuhota vastustajan yksikön ja siten välttyä vastahyökkäykseltä, joka voisi tuhota tekoälyn ohjaaman yksikön.

5 TEKOÄLYN TOTEUTUS

Opinnäytetyössä toteutetaan Planet Cube -peliprojektiin tietokonepelaajan tekoälyn päättely-arkkitehtuuri. Arkkitehtuuri pohjautuu Bergsman ja Spronckin esittelemään ADAPTA-päättelyarkkitehtuuriin. Vaikka ADAPTA-päättelyarkkitehtuuri on sellaisenaan hyvin selkeä eikä vaadi suuria määriä muutoksia, on tekoälyssä paljon tehtävää. ADAPTA-päättelyarkkitehtuurissa tekoälyn toiminnot jaetaan moduuleihin, joista jokainen toteuttaa yhtä osaa tekoälyn kokonaistoiminnasta. Näiden moduulien suunnittelu ja toteutus on yksi tämän opinnäytetyön tavoitteista. Lisäksi tavoitteena on suunnitella, kuinka ADAPTA-arkkitehtuuriin pohjautuva tekoäly implementoidaan peliin. Peliä kehitetään Unity3D-pelimoottorilla ja C#-ohjelmointikielellä ja toteutettavan tekoälyn tulisi toimia tässä ympäristössä.

5.1 Planet Cube -pelin esittely

Planet Cube on vuoropohjainen kolmiulotteinen strategiapeli PC-tietokoneille. Videopelissä pelaajat kamppailevat toisiaan vastaan kuution muotoisten planeettojen pinnoilla. Planeetan pinnan muodot ja kuution reunat toimivat näkö- ja liike-esteinä ja siten tuovat peliin uuden strategisen elementin. Pelin esikuvina toimivat vuoropohjaiset strategiapelisarjat Advance Wars ja Fire Emblem, ja monet pelilliset elementit ovat tuttuja näitä sarjoja pelanneille.

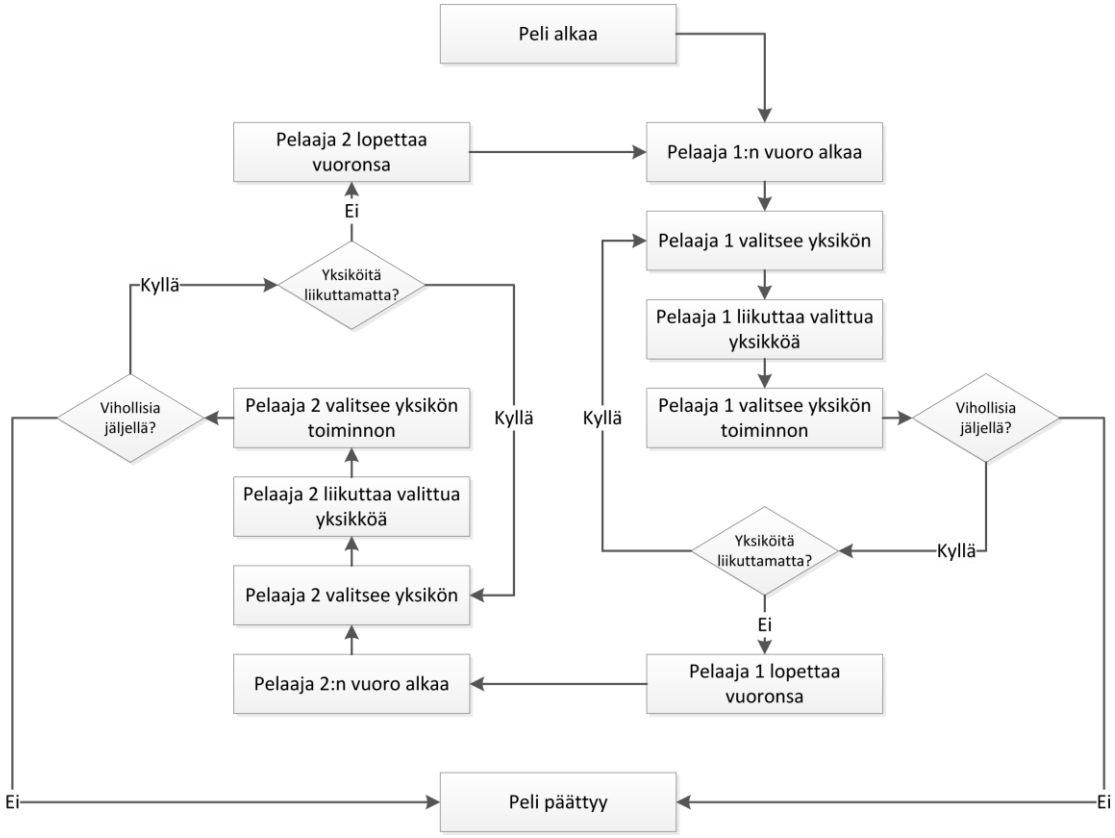
Pelin kartat ovat ruudukkoja, jotka on käärityt kuution pinnan ympärille. Ruudukoiden ruudut voivat sijaita eri korkeuksilla, ja eri korkeuksilla olevien ruutujen välillä voidaan liikkua ainoastaan ramppeja pitkin. Rampit myös mahdollistavat maata pitkin liikkuvien yksiköiden liikkumisen kuution sivuilta toisille. Lentävät yksiköt voivat liikkua erikorkuisilta ruuduilta toisille ilman ramppeja, ja ne eivät tarvitse ramppeja sivulta toiselle liikkumiseen. Kuviossa 2 voidaan nähdä nämä pelissä esiintyvät maaston muodot ja ominaisuudet.

Kuvio 2: Planet Cube -pelin kehitysversiosta otettu kuvankaappaus.

Peli tukee tällä hetkellä kahden pelaajan välisiä otteluita, ja vastapelaajaa voi edustaa toinen ihmispelaaja tai opinnäytetyön aikana kehitettävän tekoälyn ohjaama tietokonepelaaja. Lisäksi peliin on tarkoitus myöhemmässä vaiheessa kehittää yksinpelattava kampanja, jossa pelaaja kamppailee tietokoneen ohjaamia joukkoja vastaan erilaisissa tehtävissä.

5.1.1 Pelin kulku

Planet Cube -pelissä yksittäinen pelikerta on jaettu vuoroihin. Pelaajat pelaavat vuoronsa vuorotellen ja toisen pelaajan suorittaessa vuoroaan toinen pelaaja odottaa oman vuoronsa alkamista. Kuviosta 3 käy ilmi, kuinka peli etenee.



Kuvio 3: Planet Cube -pelin kulkua kuvaava vuokaavio.

Peli alkaa ensimmäisen pelaajan vuorolla. Vuoronsa alussa pelaaja valitsee yhden käytössään olevista yksiköistä napauttamalla yksikköä hiirellään. Tekoälyn ohjaama pelaaja ei käytä hiirtä yksikön valitsemiseen, mutta tekoälypelaajan käytössä on sama ohjausrajapinta, jota ihmispelaajakin käyttää. Yksikön valinnan jälkeen peli vaihtuu reitinvalintatilaan, jossa pelaaja voi valita reitin, jota pitkin valittu yksikkö kulkee. Kun pelaaja on valinnut haluamansa reitin, hän voi hyväksyä reitin klikkaamalla ruutua, johon yksikkö lopuksi pysähtyy. Yksikön siirtymiset ruudusta toiseen rekisteröidään tapahtumaketjuun, ja jokainen tapahtuma pitää sisällään tiedon siitä, mikä yksikkö on siirtynyt sekä tiedon lähtö- ja kohderuudusta. Yksiköille on määritelty liikemäärä, joka määrittää sen, kuinka monta ruutua yksikkö voi vuoron aikana siirtyä.

Yksikön siirryttyä uuteen paikkaan, tai jäätyään paikalleen, valitaan yksikön käyttämä kyky. Yksikön kyvyillä tarkoitetaan erilaisia toimintoja, joita pelaaja voi liikutetulla yksiköllä tehdä. Yksikön käytössä olevat kyvyt ovat yksikkötyyppikohtaisia, mutta seuraavat peruskyvyt ovat aina olemassa:

a) Odottaminen, joka tarkoittaa sitä, että yksikkö jää odottamaan pelaajan seuraavaa vuoroa tekemättä mitään.

b) Hyökkäys, joka tarkoittaa sitä, että yksikkö hyökkää jonkin pelaajan yksikön hyökkäyskantamalla olevan kohteen kimppuun.

Lisäksi yksiköillä voi olla muita kykyjä, joiden ominaisuudet vaihtelevat suuresti. Suurin osa toiminnoista asettaa yksikön ”käytetty”-tilaan ja yksikkö on käytettävissä uudestaan vasta pelaajan seuraavalla vuorolla. Ennen käytettävän kyvyn valitsemista pelaaja pystyy halutessaan kelaamaan yksikön vuoron taaksepäin yksikön valintaan asti. Tekoälyn ohjaamalle pelaajalle tällä toiminolla ei ole käyttöä, mutta ihmispelaajalle tämä mahdollistaa yksiköiden liikkeiden suunnittelun ilman, että pelaajan päätökset olisivat peruuttamattomia. Joissain tapauksissa yksikön vuoroa ei voida kelata taaksepäin ja pelaajan on pakko valita kyky, jota yksikkö käyttää. Esimerkiksi voidaan ottaa tilanne, jossa yksikkö on liikkunut alueelle, joka on taistelukenttää peittävän usvan peitossa. Jos usvasta paljastuu vastustajan yksiköitä, yksikkö tulee yllätetyksi ja ei ole kykenevä peruuttamaan liikettään. Tällaisissa tilanteissa yksikön täytyy jäädä paikalleen odottamaan tai käyttää jotain muuta kykyä.

Yksikön liikkumisen ja kyvyn valinnan jälkeen pelaaja voi pelata seuraavan yksikkönsä. Pelaajan siirrettyä kaikki yksikkönsä pelaajalla ei ole enää mitään tehtävää ja pelaajan vuoro päättyy. Pelaaja voi myös vaihtoehtoisesti lopettaa vuoronsa kesken, vaikka kaikkia yksiköitä ei ole liikutettu. Yksiköiden liikuttaminen ei ole pakollista, ja vuoron voi lopettaa ilman ainuttakaan siirtoa.

Pelaajan lopetettua vuoronsa vuoro siirtyy seuraavalle pelaajalle. Vuoron alussa vuoroon tullut pelaaja näkee yhteenvedon edeltävän pelaajan vuorosta (tai mahdollisessa yli kahden pelaajan pelissä, edeltävien pelaajien vuoroista). Yhteenvedossa näytetään kaikki ne vastustajan tekemät siirrot, jotka ovat vuoroon tulleen pelaajan nähtävissä. Yhteenvedon jälkeen vuorossa oleva pelaaja pelaa vuoronsa aikaisemmin kuvatulla tavalla. Tämä jatkuu, kunnes vain yhdellä pelaajalla on yksiköitä jäljellä tai yksi pelaajista saavuttaa pelimuodon muut tavoitteet, mikäli sellaisia on.

5.1.2 Karttaruudut

Planet Cube -pelin pelikenttä koostuu ruudukosta, jonka yksittäisten ruutujen arvot vaikuttavat pelin kulkuun. Karttaruudulla on seuraavat arvot, jotka määräytyvät ruudun tyypin mukaan:

a) Avoidance-arvo lisää ruudussa seisovan yksikön hyökkäyksen väistämisen todennäköisyyttä. Mitä suurempi arvo on, sitä todennäköisempää yksikköön kohdistuvan hyökkäyksen väistäminen on. Arvon ollessa nolla yksikön hyökkäyksen väistämisen todennäköisyys on se, mikä yksikön arvoihin on määritelty.

b) Defence-arvo lisää ruudussa seisovan yksikön puolustusta. Mitä suurempi arvo on, sitä enemmän yksikköön kohdistuvasta vahingosta poistetaan. Arvon ollessa nolla yksikön puolustus on se, mikä yksikön arvoihin on määritelty.

Lisäksi jokaiselle ruututyypille on määritelty, kuinka haastavaa ruutuun siirtyminen on. Tällä tarkoitetaan sitä, kuinka paljon yksikön liikemäärästä vähennetään, kun yksikkö astuu ruutuun. Yksiköille on neljä erilaista liiketyyppiä, ja tämä arvo voi olla eri yksikön liiketyypistä riippuen.

5.1.3 Defeat-pelimuoto

Defeat-pelimuodossa pelaajan tulee tuhota kaikki vihollisen yksiköt pelin voittamiseksi. Pelimuodossa ei ole mitään muita tavoitteita. On mahdollista, että peliin kehitetään elementtejä, joista pelaajan yksiköt saavat hyödykkeitä. Tällaiset elementit voisivat olla esimerkiksi pelissä olevat ruudut, joihin asettuvat yksiköt saavat jonkinlaisia hyödykkeitä. Nämä hyödykkeet voisivat olla esimerkiksi:

a) Liikemäärän kasvaminen. Yksikön liikemäärä kasvaa jollain ennalta määritetyllä määrällä muutaman vuoron ajaksi.

b) Hyökkäysvoiman kasvaminen. Yksikön hyökkäysvoima kasvaa jollain ennalta määritellyllä määrällä muutaman vuoron ajaksi.

c) Muut vastaavat lisät.

Tällaiset lisäelementit tarjoavat pelaajille vaihtoehtoisia tavoitteita, joiden saavuttaminen ei vielä päätä peliä. Tämä lisäisi peliin erilaisia lisätavoitteita monimutkaistamatta lopullista voittoehtoa. Tämän kaltaiset lisäelementit olisivat käytössä kaikissa pelimuodoissa ja toisivat peliin lisää strategista syvyyttä. Näiden tavoitteiden tavoittelu lisäisi tekoälyn toimintoja ja tästä syystä ADAPTA-päättelyarkkitehtuurin laajennettavuus tulee tarpeeseen.

5.2 Vaatimusmäärittely

Planet Cube -peli asettaa tekoälylle joitain vaatimuksia siitä, millaisia tehtäviä tekoälyn tulisi kyetä suorittamaan. Nämä tehtävät ovat sellaisenaan irrallisia ADAPTA-päättelyarkkitehtuurin tasoista eikä niiden tarkoituksena ole määritellä tekoälyn moduuleita. Sen sijaan tarkoituksena on määrittää ne tehtävät, jotka tekoälyn tulee toteuttaa joko strategisella tai taktisella tasolla tai näiden tasojen toimintojen yhdistelmänä. Tällaisia tehtäviä ovat esimerkiksi omien yksiköiden suojaaminen ja vihollisten yksiköiden tuhoaminen.

5.2.1 Tiedustelutoiminta

Planet Cube -pelissä tekoälyn ohjaamalla pelaajalla ovat käytössään samat tiedot, jotka samaa sotajoukkoa ohjaavalla ihmispelaajalla olisi. Tämä tarkoittaa sitä, että mikäli vihollisen yksiköt eivät näe pelaajan yksiköitä, ei tekoälypelaaja tiedosta näiden yksiköiden olemassa oloa, vaikka teknisesti olisi täysin mahdollista kirjoittaa tekoäly, joka hakee tiedot tietokonejärjestelmästä. Tekoäly toteutetaan tällä tavalla, jotta pelaaja ei tuntisi oloaan huijatuksi. Tästä syystä yksi tekoälyn merkittävimmistä toiminnoista on tiedustelu.

Tiedustelulla tekoälypelaaja hankkii tietoa vihollisen yksiköiden sijainneista ja niiden määrästä. Kerätty tieto toimii pohjana suurelle osalle muita toimintoja. Jos tekoälypelaaja ei tiedä vihollisyksiköiden sijainteja, pyrkii se siirtämään omia yksiköitään sellaisiin paikkoihin, joista kartoitettaisiin mahdollisimman paljon näkymättömissä olevia alueita. Tiedustelemalla kerätty tieto tallennetaan muun muassa vaikutuskarttoihin myöhempää käyttöä varten.

5.2.2 Hyökkääminen

Strategiapeleissä, joissa päätavoitteena on tuhota vihollisen yksiköt taistelussa, täytyy tekoälyn kyetä toimimaan hyökkäävästi. Hyökkäävällä toiminnalla tarkoitetaan kaikkia niitä siirtoja, jotka tekevät vahinkoa vihollisen yksiköihin tai pyrkivät siirtämään omia yksiköitä sellaisiin sijainteihin, joista vahinkoa voitaisiin tuottaa. Hyökkäävien siirtojen suunnittelussa käytetään avuksi tiedustelemalla kerättyä tietoa vihollisyksiköiden nykyisistä ja menneistä sijainneista.

5.2.3 Yksiköiden suojaaminen

Tekoälyn ohjaaman pelaajan täytyisi pystyä myös suojaamaan omia yksiköitään. Tekoälyn omien yksiköiden suojaamiseen tehokkain tapa olisi vältellä taistelutilanteita. Tämä ei kuitenkaan olisi kovin järkevää ottaen huomioon, että Defeat-pelimuodon päätavoite on vihollisen yksiköiden tuhoaminen ennen kuin omat yksiköt ovat tuhoutuneet. Tästä syystä tekoälyn tulisi kyetä suojaamaan omia yksiköitään nimenomaan taistelutilanteissa. Tekoälyn tulisikin pystyä arvioimaan kuinka suuri hyöty oman yksikön uhraamisesta olisi verrattuna yksikön menettämisestä johtuviin haittoihin. Mikäli haitat ylittävät saadut hyödyt, tulisi tekoälyn siirtää yksikkö suojaan ja pyrkiä minimoimaan yksikköön kohdistuva vahinko.

5.3 Päätöksentekoon vaikuttavat muuttujat

Tekoälyn toiminnan ja sen tehokkuuden takaamiseksi tekoälyn toiminta pitäisi olla sidoksissa pelimaailman sen hetkiseen tilaan. Koko pelimaailman tarkkaileminen on tehtävänä liian suuri, jotta tekoäly kykenisi tehokkaasti keräämään ja hyödyntämään pelimaailman tilaan liittyvää tietoa. Tästä syystä on erittäin tärkeä määrittää pelimaailmasta ne asiat, joilla on vaikutusta tekoälyn tekemiin päätöksiin. Näitä asioita kutsutaan päätöksentekoon vaikuttaviksi muuttujiksi. Seuraavaksi esitellään kaikki päätöksentekoon vaikuttavat muuttujat, kerätyn tiedon talletiointimuodot sekä muuttujien käyttökohteet.

5.3.1 Tekoälyn omistamien yksiköiden elämäpisteet

Tekoälyn tekemiin päätöksiin vaikuttavat myös yksiköiden elämäpisteet. Elämäpisteillä tarkoitetaan lukua, joka kuvastaa sitä, kuinka paljon yksikkö voi vastaanottaa vahinkoa. Kun yksikköön kohdistuu vahinkoa, vahinko vähennetään yksikön elämäpisteistä. Mikäli yksikön elämäpisteet laskevat nollaan tai sen alle, yksikkö luetaan kuolleeksi ja se poistetaan pelikentältä.

Elämäpisteet vaikuttavat tekoälyn toiminnassa päätöksiin, jotka mahdollistavat omien yksiköiden vahingoittumisen. Tällaisia tilanteita ovat esimerkiksi taistelu- ja liikkumistilanteet. Yksiköiden elämäpisteet vaikuttavat myös siirtojen suoritusjärjestykseen. Tekoälyn ei esimer-

kiksi ole kannattavaa siirtää ensimmäisenä hyökkäysvuoroon yksikköä, joka vastahyökkäyksen sattuessa kuolisi mahdollisesti itse, tai siirtää yksiköitä vihollisen uhkaamille alueille mikäli on mahdollista, että yksikkö tulisi tuhoutumaan siirron seurauksena.

5.3.2 Yksiköiden strateginen arvo

Aivan kuten oikeassakin sodankäynnissä, myös strategiapeleissä yksiköt ovat strategisesti eriarvoisia. Tällä tarkoitetaan sitä, kuinka hyödyllinen yksikkö on sen omistavalle osapuolelle osana jotakin suurempaa tavoitetta. Esimerkiksi shakissa pelinappuloille on määritelty arvot, jotka korreloivat hyvin pitkälti pelinappulan sääntöjen määrittelemän liikkumistavan mukaan. Tässä poikkeuksena on kuitenkin kuningas, jonka menettäminen tarkoittaisi automaattisesti pelin päättymistä ja siten se on pelin arvokkain nappula. Planet Cube -pelissä yksiköiden arvo on kuitenkin riippuvainen vihollisjoukkueen kokoonpanosta, joten mitään absoluuttisia arvoja yksiköille ei voida määrittää. Yksikön strategista arvoa ei tule sekoittaa resurssien huutokauppahetkellä määritettyyn hyötyarvoon.

Yksiköiden arvot vaikuttavat tekoälyn toiminnassa siihen, kuinka tarkasti mitäkin yksiköitä suojellaan mahdollisilta riskeiltä ja suhteutetaan riskit saatuihin hyötyihin. Esimerkiksi tekoälylle ei ole kannattavaa siirtää yksikköä, jonka olemassaolo takaa ottelun voiton, alueelle, jolla yksikkö joutuisi hyvin suureen vaaraan. Yksiköitä tulee kuitenkin käyttää tilanteissa, joissa yksikön strategista arvoa voidaan hyödyntää.

5.3.3 Vihollisjoukkueen kokoonpano

Kuten monissa muissakin strategiapeleissä, myös Planet Cube -pelissä yksiköiden välillä vallitsee heikkous- ja vahvuussuhteita. Tällä tarkoitetaan sitä, että jokin yksikkö on selkeästi parempi yhdenlaisia yksiköitä vastaan ja selkeästi heikompi toisenlaisia yksiköitä vastaan. Tietoa tällaisista voimasuhteista ja vihollisjoukkueen kokoonpanosta voidaan käyttää hyödyksi tekoälyn päätöksen teossa strategisella tasolla. Vertailemalla omien yksiköidensä voimasuhteita vihollisen omistamiin yksiköihin voi tekoäly tehdä arvioita siitä, minkä tyyppiset yksiköt ovat tekoälylle tärkeitä voittoa tavoitellessa. Tällä tavoin tekoäly pystyy paremmin arvioimaan tekemänsä ratkaisun kannattavuutta.

5.3.4 Vihollisyksiköiden sijainnit

Ottaen huomioon Defeat-pelimuodon tavoitteen, yksi tärkeimpiä pelimaailmasta kerättäviä tietoja ovat vihollisyksiköiden sijainnit pelimaailmassa. Tämän tiedon pohjalta voidaan arvioida, minne liikkumalla voidaan uhata ja mahdollisesti tuhota vihollisen yksiköitä sekä mitkä omista yksiköistä ovat uhattuna vihollisen yksiköiden toimesta. Tämän tiedon kerääminen voitaisiin teknisesti toteuttaa keräämällä tieto suoraan pelisovelluksen tietorakenteista, mutta tämä olisi epäreilua pelaajaa kohtaan. Jotta pelaaja ei tuntisi tulleensa huijatuksi, myös tekoälyn tulee kerätä tieto vihollisyksiköiden sijainneista samalla tapaa kuin pelaajan eli liikuttelemalla omia yksiköitä ja siten paljastaa pelimaailmaa sitä peittävästä usvasta.

Tekoäly tallentaa tekoälyn näkyvyysalueella olevat vihollisyksiköt listaan. Listassa oleva yksikkö poistetaan, jos yksikkö katoaa tekoälypelaajan yksiköiden näkökentästä. Tätä listaa näkyvillä olevista vihollisyksiköistä ja niiden sijainneista hyödynnetään luodessa tietoa, joka pohjautuu vihollisyksiköiden sijainteihin. Tämän lisäksi tekoäly säilöö tiedon vihollisista vaikutuskarttaan.

5.3.5 Vihollisyksiköiden hyökkäysalueet

Vihollisyksiköiden hyökkäysalueet tallennetaan jokaista näkyvää vihollisyksikköä kohden ruutulistauksena, johon listataan kaikki ruudut, joihin yksikkö pystyy hyökkäämään seuraavalla vuorolla. Lisäksi yksikkökohtaisista hyökkäysalueista koostetaan yleinen hyökkäysalue, johon listataan kaikki ne ruudut, joihin näkyvät vihollisen yksiköt voivat hyökätä seuraavalla vuorolla. Näitä tietoja voidaan käyttää siirtoja suunniteltaessa selvittämään, mitkä alueet ovat yleisesti ottaen hyökkäysuhan alla sekä arvioimaan hyökkäävien yksiköiden luoman uhan vakavuutta hyökkäysalueelle siirtoa suunnitellessa.

Vihollisyksiköiden hyökkäysalueet ovat faktatietoa ja käsittelevät juuri tämän hetkistä pelitilannetta. Tällaisen tiedon käyttäminen ennusteiden ja ajallisesti kattavamman arvioinnin tekemiseen on työlästä, ja tähän tarkoitukseen tekoäly käyttää uhka-aluearviointia.

5.3.6 Uhka-alueet

Uhka-alueilla tarkoitetaan tekoälyn käyttämää tietoa, jossa vaikutuskarttoihin tallennettuna säilötään tämän hetkisiä ja menneitä vihollisyksiköiden hyökkäysalueita. Vaikutuskarttaan säilöttynä hyökkäysalueet saavat uuden ulottuvuuden, ajan. Kun vaikutuskarttaan säilöttyjä vaikutuksia heikennetään ajan kuluessa, tieto hyökkäysalueista ei ole enää luonteeltaan binääristä vaan tiedolla on useita tiloja. Vaikutuskartassa olevan vaikutuksen voimakkuus kertoo osittain siitä, kuinka kauan sitten alue oli uhattuna.

Uhka-alueita hyödynnetään tekoälyn yksiköiden liikkeitä suunniteltaessa, kun on tarpeellista tietää millä alueilla vastustajan yksiköt ovat olleet menneisyydessä tai ovat juuri tällä hetkellä. Tämä tieto on tarpeellista esimerkiksi suunniteltaessa siirtoja, joiden tarkoituksena on viedä omia yksiköitä hyökkäysasemiin lähemmäs vastustajan yksiköitä. Koska uhka-alueet säilövät myös mennyttä tietoa, voivat yksiköt hakeutua alueille, joilla vastustajan yksiköitä on ollut, vaikka yksiköt eivät olisikaan juuri nyt havaittavissa. Uhka-aluetietoa hyödynnetään myös silloin, kun halutaan suojella jotakin tärkeää resurssia. Tällaisia resursseja voisivat olla esimerkiksi yksiköt, joiden turvaaminen on erityisen tärkeää pelin voittamisen kannalta.

5.4 Moduulien suunnittelu

Tekoälyn päättelyarkkitehtuuri koostuu siis kahdesta tasosta, joilla toimii tekoälyn toiminnasta vastaavia moduuleita. Seuraavaksi esitellään kaikki näillä tasoilla toimivat moduulit. Lisäksi käsitellään näiden moduulien toimintatavat ja se, millä tavoin moduulit vaikuttavat tekoälyn toimintaan.

5.4.1 Intelligence-moduuli

Intelligence-moduuli on strategisella tasolla toimiva moduuli, jonka tehtävänä on kerätä ja ylläpitää tiedustelutietoa, jota tekoälyn taktisella tasolla toimivat moduulit käyttävät hyödykseen suunnitellessaan tulevia siirtoja. Jokaisen vuoron alussa tekoäly aloittaa toimintansa tästä moduulista. Ensimmäisenä moduuli kerää tarvittavat tiedot pelimaailmasta ja päivittää tämän hetkistä pelin tilaa käsittelevät tiedot. Intelligence-moduuli pitää kirjaa seuraavista asioista:

vihollisjoukkueen kokoonpano, vihollisyksiköiden sijainnit, vihollisyksiköiden hyökkäysalueet ja uhka-alueet. Kun moduuli on suorittanut toimintansa loppuun ja saanut tiedustelutiedot päivitettyä, luovuttaa se toimintavuoron Asset Allocator -moduulille.

5.4.2 Asset Allocator -moduuli

Asset Allocator -moduuli on strategisella tasolla toimiva moduuli, jonka tarkoitus on pitää kirjaa tekoälyn ohjaaman pelaajan hallitsemista resursseista, tässä tapauksessa yksiköistä, ja tarjota näitä resursseja taktisen tason moduulien käytettäväksi.

Moduuli toimii seuraavalla tavalla. Se kartoittaa kaikki tekoälyn hallinnassa olevat yksiköt ja karsii niiden joukosta pois kaikki ne, jotka ovat syystä tai toisesta tämän hetkisellä vuorolla toimintakyvyttömiä. Kun yksiköt on valikoitu, antaa Asset Allocator -moduuli yksiköt taktisen tason moduulien käsiteltäväksi huutojen luomista varten. Kun taktisen tason moduulit ovat luoneet huutonsa, palauttavat ne huudot Asset Allocator -moduulille. Tämän jälkeen Asset Allocator -moduuli luovuttaa toimintavuoron Utility Management -moduulille ja toimittaa samalla taktisen tason moduulien luomat huudot Utility Management -moduulin arvioitaviksi.

5.4.3 Utility Management -moduuli

Utility Management -moduuli on strategisella tasolla toimiva moduuli, jonka vastuulla on taktisen tason moduulien luomien resursseja koskevien huutojen hyödyllisyyden arvioiminen. Kun taktisen tason moduulit ovat luoneet huudot tarjolla olevista resursseista, Utility Management -moduuli pyrkii allokoimaan resurssit parhaalla mahdollisella tavalla siten, että resurssien allokointi hyödyttää kaikkia moduuleja mahdollisimman paljon ja on linjassa tekoälyn strategisten tavoitteiden kanssa.

Huutojen arvioimisen jälkeen Utility Management -moduuli antaa resurssit haluamiensa taktisen tason moduulien käytettäväksi. Taktisella tasolla toimivat moduulit luovat siirtoja käyttöönsä saamilleen resursseille. Moduulit palauttavat luodut siirrot Utility Management -moduulille, joka toimittaa siirrot eteenpäin Unit Protection -moduulille. Tämän jälkeen toimintavuoro siirtyy Unit Protection -moduulille.

5.4.4 Unit Protection -moduuli

Yksiköitä suojaavan Unit Protection -moduulin päätarkoituksena on karsia Utility Management -moduulin toimittamista siirroista pois sellaiset siirrot, jotka vaarantavat yksiköitä ilman merkittävää hyötyä tai vaarantavat yksiköitä, joiden menettäminen olisi haitallista siirron tuomista hyödyistä huolimatta. Unit Protection -moduuli toimii tekoälyn strategisella tasolla.

Moduulin toiminnan pohjautuessa toteutettavien siirtojen karsimiseen eikä siirtojen muokkaamiseen on vaarana, että vuoron aikana käytettävissä olevia resursseja jää käyttämättä. Moduulin toteutus oli osittain ADAPTA-päättelyarkkitehtuurin ideologian sanelemaa, ja tästä aiheutuvista ongelmista olen kirjoittanut jatkokehityskohteita käsittelevässä luvussa.

Kun siirroista on poistettu kaikki huonot ja kannattamattomat siirrot on Unit Protection -moduulin työ tehty. Tämän jälkeen moduuli luovuttaa toimintavuoron Movement Order -moduulille. Myös hyväksytyt siirrot siirretään eteenpäin seuraavan moduulin käsiteltäviksi.

5.4.5 Movement Order -moduuli

Strategisella tasolla toimivan Movement Order -moduulin tehtävänä on järjestellä taktisen tason moduulien luomat ja Unit Protection -moduulin hyväksymät siirrot parhaaseen mahdolliseen suoritusjärjestykseen. Tällä järjestelytyöllä pyritään minimoimaan omiin yksiköihin aiheutuva vahinko ja maksimoimaan vihollisen yksiköihin tehtävä vahinko.

Movement Order -moduuli kartoittaa moduulille toimitetuista siirroista niitä, jotka jakavat tietoa, ja ratkaisee seuraavat tilanteet:

a) Mikäli hyökkäävillä siirroilla on sama kohdeyksikkö, lasketaan siirtojen yhteenlaskettu vahingon määrä. Jos vahingon määrä ylittää kohdeyksiköllä jäljellä olevat elämäpisteet, hyökkäävät siirrot pyritään järjestelmään siten, että vähiten vahinkoa vastahyökkäyksestä ottavat yksiköt hyökkäävät ensin, ja eniten vahinkoa ottavat viimeisenä. Lisäksi hyökkäysvuorot järjestellään hyökkäävillä yksiköillä olevien elämäpisteiden mukaan.

b) Tilanteessa, jossa useita yksiköitä on siirtymässä vuoron aikana sivulle, jolla tekoälyn omistamia yksiköitä ei ole, Movement Order -moduulin täytyy järjestellä siirrot par-

haaseen mahdolliseen järjestykseen. Movement Order -moduuli arvioi yksiköiden strategisia arvoja, yksiköiden kykyä kestää vahinkoa ja uhka-aluetietoa ja näiden tietojen pohjalta järjestelee siirrot turvallisimpaan mahdolliseen järjestykseen.

c) Mikäli kaksi tekoälyn yksikköä on suorittamassa siirtoja vuoron aikana ja toisen yksikön siirto vaatii onnistuakseen toisen yksikön säilyttävän sen paikan, jossa se vuoron alkaessa oli, täytyy Movement Order -moduulin varmistaa, että siirrot tullaan ajamaan sellaisessa järjestyksessä, että tämä onnistuu. Esimerkiksi tilanteessa, jossa ensimmäinen yksikkö on aikeissa parantaa toisen yksikön elämäpisteitä, tulee ensimmäisen yksikön saada suorittaa parannustoimenpide ennen kuin toinen yksikkö saa suorittaa oman siirtonsa.

Kun Movement Order -moduuli on saanut järjesteltyä siirrot sopivaan järjestykseen, siirtyvät siirrot suoritettavaksi. Suoritusvaiheessa tekoälyn toteuttama ja järjestelemä siirtoketju käsitellään ja siirrot suoritetaan siinä järjestyksessä, missä ne siirtoketjussa ovat. Tämän jälkeen pelivuoro luovutetaan seuraavalle pelaajalle ja tekoälyn työ vuoron osalta on valmis.

5.4.6 Siirtomoduuli

Tekoälyn taktisella tasolla toimivan siirtomoduulin tavoitteena on siirtää tekoälyn ohjaamia yksiköitä taisteluun ja pois taistelusta. Moduulin toiminta pohjautuu hyvin paljon uhka-aluetiedon tarkasteluun. Moduulilla on seuraavat kaksi päätehtävää:

a) Siirtää taistelemaan kykeneviä yksiköitä kentällä sellaisille alueille, joilla vihollisten tiedetään olleen tai olevan juuri tällä hetkellä. Tällä tavoin omat yksiköt saadaan lähemmäksi vihollisen yksiköitä ja hyökkäysmoduulin käyttöön.

b) Siirtää heikossa kunnossa tai vaarallisessa tilanteessa olevia yksiköitä pois vihollisen uhkaamilta alueilta. Tämän toiminnan tarkoituksena on suojella yksiköitä uhkilta, jotka aiheutuvat pelaajan toimista riippumattomista syistä. Vaikka strategisella tasolla toimiva Unit Protection -moduuli suojelee yksiköitä tekoälyn moduulien tekemiltä virheiltä, ei kyseinen moduuli ole kykenevä reagoimaan vihollisyksiköiden hyökkäysyrityksiin ja muihin vastaaviin uhkiin.

Moduuli määrittelee yksiköille hyötyarvon sen mukaan, kuinka tärkeänä tai kiireellisenä moduuli pitää yksikön saamista uhka-alueelle tai pois sieltä. Korkeimman hyötyarvon resursseja ovat yksiköt, jotka seisovat vihollisyksikön hyökkäysalueella ja joiden elämäpistemäärä on merkittävän alhainen. Matalimman hyötyarvon resursseja ovat yksiköt, jotka ovat siirtymässä uhka-aluetta kohti, mutta siirrosta ei ole vielä välitöntä hyötyä.

Kun siirtomoduuli saa resurssin käytettäväkseen, moduuli pyrkii kartoittamaan pelistä seuraavia tilanteita ja kehittämään ratkaisun niihin:

a) Jos jokin yksiköistä on välittömässä hengenvaarassa, siirtomoduuli pyrkii löytämään yksikölle turvallisen pakoreitin pois vaara-alueelta. Mikäli mahdollisuutta vaara-alueelta pois siirtymiseen ei ole, moduuli pyrkii löytämään vaara-alueelta ruudun, jonka arvot pienentävät mahdollisesti saadun vahingon määrää tai vähentävät vihollisen osumamahdollisuutta.

b) Jos jollakin uhka-alueella on tekoälyn omistamia yksiköitä taistelemassa vihollisyksiköitä vastaan, moduuli arvioi kyseisen taistelun voimasuhteet ja pyrkii tarvittaessa siirtämään lisäyksiköitä omien yksiköiden tueksi taisteluun.

c) Jos jokin taistelukunnossa oleva yksikkö on alueella, joka ei ole vihollisyksiköiden läheisyydessä, pyrkii tekoäly siirtämään tätä yksikköä lähemmäksi vihollisen uhkaamia alueita.

5.4.7 Hyökkäysmoduuli

Taktisella tasolla toimivan hyökkäysmoduulin päätavoitteena on aiheuttaa vihollisjoukkueelle mahdollisimman paljon vahinkoa ja tuhota niin monta vihollisyksikköä kuin mahdollista. Mikäli jokin moduulille tarjotuista yksiköistä on hyökkäysetäisyyden päässä vihollisen yksiköstä, hyökkäysmoduuli käsittelee yksikön käyttökelpoisena resurssina ja luo kyseistä yksiköstä huudon. Yksiköiden hyötyarvo perustuu seuraaviin asioihin:

a) Hyökkäävän yksikön ja kohteen välinen voimasuhde vaikuttaa hyötyarvoon siten, että mitä tehokkaampi hyökkäävä yksikkötyyppi on kohdeyksikköä vastaan, sitä korkeamman hyötyarvon hyökkäävä yksikkö saa.

b) Mitä useampia mahdollisia hyökkäyskohteita tekoälyn ohjaamalla yksiköllä on, sitä korkeammaksi yksikön hyötyarvo arvioidaan.

c) Mikäli yksikön hyökkäysetäisyydellä on vihollisyksiköitä, jotka tulisivat todennäköisesti tuhoutumaan hyökkäyksen seurauksena, yksikkö saa korkeamman hyötyarvon. Tällä tavoin yksiköt, jotka voisivat vähentää vihollisjoukkueen hyökkäysvoimaa tuhoamalla vihollisen yksiköitä, pääsevät todennäköisemmin hyökkäämään vuoron aikana. Tämä vähentää tulevaisuudessa vihollisjoukkueen aiheuttamaa uhkaa.

Saatuaan resurssit käytettäväkseen hyökkäysmoduuli luo parhaat mahdolliset siirrot yllä mainittujen säännösten pohjalta. Mikäli yksiköllä on hyökkäysetäisyydellään useampia kuin yksi vihollisen yksikkö, täytyy tekoälyn valita kohteista paras. Tässä tapauksessa ensisijaisia kohteita ovat vihollisyksiköt, jotka tuhoutuvat hyökkäyksestä, ja toissijaisia kohteita ovat yksiköt, joihin tekoälyn yksikkö aiheuttaa runsaasti vahinkoa.

5.4.8 Tiedustelumoduuli

Taktisella tasolla toimivan tiedustelumoduulin päätavoite on saada mahdollisimman paljon tietoa vihollispelaajan toimista. Tiedustelumoduuli kerää tietoa vihollisyksiköiden sijainneista ja vihollisjoukkueen kokoonpanosta siirtämällä omia yksiköitään sellaisiin ruutuihin, joista yksiköt saavat pelikenttää näkyviin mahdollisimman paljon. Tiedustelumoduulin tarkoituksena ei ole itsenäisesti hallita ja tallentaa tiedustelutietoa, vaan sen sijaan pyrkiä paljastamaan vihollisen toimet strategisella tasolla toimivalle Intelligence-moduulille. Tiedustelumoduulin määrittelemät yksiköiden hyötyarvot perustuvat seuraaviin tekijöihin:

a) Yksiköt, joiden liikenopeus on suuri, saavat hyötyarvoon korotuksen, koska näillä yksiköillä voidaan siirtyä nopeasti paikasta toiseen jopa pitkienkin etäisyyksien päähän.

b) Yksiköt, joiden näköetäisyys on suuri, saavat hyötyarvoon korotuksen, koska näillä yksiköillä voidaan kartoittaa suuria alueita helpommin kuin lyhemmän näköetäisyyden omaavilla yksiköillä.

c) Mikäli yksiköllä on pääsy alueille, jotka eivät ole tekoälyn ohjaaman joukkueen näköalueella, yksikkö saa hyötyarvoon korotuksen. Jos näistä joukkueen näköalueen ulko-

puolisista alueista ei ole vaikutuskarttaan tallennettua tietoa Intelligence-moduulin hallussa, pidetään näille alueille pääsemistä prioriteettina ja hyötyarvo on vieläkin korkeampi.

Saatuaan yksiköt käyttöönsä moduuli kartoittaa yksikön liikkumisalueelta sellaiset ruudut, jotka ovat tekoälyn ohjaaman joukkueen näköalueen ulkopuolella. Tämän jälkeen tekoäly valitsee näistä ruuduista sellaiset, joilla vihollisjoukkueen uhka on matala. Nämä alueet ovat sellaisia, joissa tekoäly ei tiedä vihollisten käyneen tai vihollisten vierailusta alueella on kulunut jo jonkin aikaa. Yksiköitä pyritään siirtämään näille alueille.

5.5 Tekoälyn toteutus

Opinnäytetyön aikana toteutettu tekoäly ohjelmoitiin, kuten Planet Cube -pelikin, C#-ohjelmointikielellä. Tähän ratkaisuun päädyttiin parhaan mahdollisen yhteensopivuuden takaamiseksi. Lisäksi C#-ohjelmointikieli tarjosi kaikki tarvittavat toiminallisuudet tekoälyn toteuttamista varten, joten ei ollut mitään tarvetta siirtyä toiseen ohjelmointikieleen.

5.5.1 Arkkitehtuuri

Opinnäytetyön aikana toteutetun tekoälyn arkkitehtuurissa on pyritty huomioimaan ADAPTA-päättelyarkkitehtuurin tavoite helposta laajennettavuudesta. Kuten liitteessä 1 on esitetty, tekoälyn pohjalle on toteutettu useita luokkia ja tietotyyppejä, joita käyttämällä tekoälyn erilaiset toiminnot oli helppo määritellä.

Tekoälyn toteutuksessa pyrittiin erottamaan tekoäly varsinaisista pelitoiminnoista siten, että tekoälyn päätoiminto oli luoda lista toiminnoista, joita tekoälyn ohjaaman joukkueen tulisi suorittaa. Tämän jälkeen varsinaisen pelin puolelle toteutetun osan tehtävä oli käsitellä tekoälyn luoma lista ja siirtää tekoälyn hallitsemia yksiköitä listassa olevien komentojen mukaisesti.

Tämän opinnäytetyön toteutushetkellä tekoäly pystyi ainoastaan valitsemaan yksikön, liikuttamaan yksikköä, asettamaan yksikön odottamaan seuraavaa vuoroa ja hyökkäämään yksiköllä vihollisyksiköiden kimppuun. Tämä johtui siitä, että valtaosa monimutkaisemmista peli-

toiminnoista oli vielä kehityksessä. Osa toiminnoista leikattiin opinnäytetyössä käytettävän tekoälyn toiminnoista pois, jotta työ olisi mahdollista saada valmiiksi. Tekoälyn kehityksessä otettiin kuitenkin huomioon pois jäävien toimintojen myöhemmän lisäämisen mahdollistaminen. Kuten liitteestä 1 käy ilmi, tekoälylle tällä hetkellä mahdolliset toiminnot on määritelty AdaptaActionType-enumeraatiossa. Tätä enumeraatiota ja siirtojen toteuttamisesta vastaavaa osaa laajentamalla tekoälyyn voidaan määritellä uusia toimintoja.

Tekoälyn taktisen tason moduulien luomat huudot määritellään AdaptaBid-luokan olioilla. Koska taktisen tason moduulit eivät välitä, mihin käyttötarkoitukseen mikäkin resurssi tullaan käyttämään, voidaan moduulien luomat huudot toteuttaa hyvin yksinkertaisella luokalla. Luokan oliosta käy ilmi seuraavat asiat: mitä resurssia on huudettu, resurssille arvioitu hyötyarvo ja mille moduulille resurssi luovutetaan käytettäväksi, mikäli huuto on voittava huuto.

Tekoälyn luomat toiminnot määrittelevä lista koostuu AdaptaActionSet-luokan olioista, jotka yhdistävät yksittäisen resurssin listaan AdaptaAction-luokan olioita. Tällaisella rakenteella pystyttiin määrittelemään toimintoryppäitä ja tarpeen tullen peruuttamaan kaikki resurssille listatut toiminnot, mikäli todettiin, ettei jotain niistä pystytty toteuttamaan syystä tai toisesta. AdaptaAction-luokan oliot määrittelevät mitä tehdään, ja AdaptaActionSet-luokan olio määrittelee tekijän.

Tekoälyn varsinaisten toimintojen pohjalle luotiin AdaptaModule-luokka, josta perittiin kaksi luokkaa lisää: AdaptaStrategyModule ja AdaptaTacticalModule. Tämä jako tehtiin, jotta strategiselle ja taktiselle tasolle toteutettavien moduulien määritteleminen olisi helpompaa ja moduuleille saataisiin luotua selkeä rajapinta, jonka avulla tekoälyn toimintoja voitaisiin lisätä ilman, että varsinaiseen tekoälyn runkoon tulisi enää tehdä muutoksia.

5.5.2 Moduulien toteutus

Yksittäiset tekoälyn moduulit perittiin AdaptaStrategyModule- tai AdaptaTacticalModule-luokasta riippuen siitä, kummalla tasolla kyseisen moduulin tuli toimia. Erilaiset moduulit pitävät sisällään moduulille olennaista tietoa ja metodeja, joiden avulla moduuli pystyy suorittamaan tarvittavat tehtävät.

Jokaisen taktisen tason moduulin tulee toteuttaa GenerateBids-metodi, jota kutsumalla strateginen taso pyytää moduuleilta huutoja listalle tarjolla olevista resursseista. Lisäksi taktisen tason moduulien tulee toteuttaa GenerateActions-metodi, jota kutsumalla strateginen taso luovuttaa resurssit niiden moduulien käyttöön, joiden huudot voittivat, ja samalla strateginen taso saa vastaukseksi moduulien luomat toiminnot kullekin resurssille. Nämä metodit määrittelevät rajapinnan, joiden avulla strateginen taso voi kommunikoida jokaisen taktisen tason moduulin kanssa identtisesti ja tekoälyn laajennettavuus yksinkertaistuu.

5.6 Tekoälyn testaus

Toteutettua tekoälyä testattiin sekä tekoälyn toiminnan takaamiseksi että mahdollisten jatkokehityskohteiden kartoittamiseksi. Opinnäytetyösuunnitelmassa määriteltiin opinnäytetyön onnistumisen arvioimiseksi seuraavat kriteerit: ”Jos tekoälyn toiminta ei ole riittävän hyvä, jotta se olisi kykenevä voittamaan testivastustajat, tulisi testien aikana kerätystä tiedosta ilmetä mitkä osat tekoälystä aiheuttavat heikon toiminnan. Tällä tapaa voidaan taata, että työstä on hyötyä pelin jatkokehityksessä, mikäli työ ei sellaisenaan ole riittävä.” Opinnäytetyöstä on siis mahdollista saada hyötyä irti, vaikka opinnäytetyön aikana toteutettu tekoäly ei olisi riittävän laadukas julkaistussa pelissä käytettäväksi.

5.6.1 Testaustavat ja testauksen tavoitteet

Opinnäytetyön aikana toteutetusta tekoälystä pyrittiin testauksen aikana selvittämään, kuinka hyvin tekoäly suoriutuu ihmispelaajia vastaan ja samalla kartoittamaan ne tekoälyn osa-alueet, jotka tarvitsevat jatkokehitystä. Lisäksi testauksen aikana pyrittiin kartoittamaan asioita, jotka heikentävät tekoälyn toimintaa, mutta eivät ole varsinaisesti osa toteutettua tekoälyä.

Testauksen aikana pelinkehitystiimin jäsenet pelasivat tekoälyä vastaan. Näiden testipelien aikana tekoälyn toimintaa tarkkailtiin ja tekoälyn tekemiä päätöksiä arvioitiin. Tekoälyn tekemiä päätöksiä seuraamalla pyrittiin kartoittamaan tekoälystä ne osat, jotka olivat osallisena kunkin päätöksen teossa. Lisäksi tällä tavoin voitiin selvittää ne tekoälyn osat, jotka vaikuttivat siihen, ettei tilanteessa toimittu tehokkaimmalla mahdollisella tavalla.

5.6.2 Ongelmat ja jatkokehityskohteet

Testauksen aikana voitiin todeta, että tekoäly suoriutui useista tilanteista suhteellisen hyvin ja oli kykenevä toimimaan jonkinlaisena vastustajana ihmispelaajalle. Oli kuitenkin ilmeistä, että tekoälyn toiminta ei ollut parasta mahdollista ja kokenut ihmispelaaja pystyi usein päihittämään tekoälyn. Testauksen päätteeksi voitiin todeta, että tekoäly ei sellaisenaan ole riittävän hyvä ollakseen osana julkaistavaa peliä ja tästä syystä on erittäin tärkeää määritellä ne ongelmakohdat, joihin tulisi jatkokehityksessä puuttua.

Merkittävimmän ongelman tekoälyn toiminnassa aiheutti ADAPTA-päättelyarkkitehtuurin rakenne, jonka pohjalta tekoälyn päättelyarkkitehtuuria lähdettiin kehittämään. ADAPTA-päättelyarkkitehtuurin modulaarinen rakenne mahdollisti tekoälyn joustavan kehittämisen ja tekoälyn toiminta oli helppo määritellä yksittäisinä moduuleina, mutta samanaikaisesti se aiheutti suuria ongelmia tekoälyn toiminnassa. Kuten ADAPTA-päättelyarkkitehtuurissa oli määritelty, yksittäiset taktisen tason moduulit ovat kiinnostuneita ainoastaan omasta toiminnastaan ja strateginen tason moduulien vastuulla on kokonaiskuvan hahmottaminen ja moduulien toiminnan järkevä yhdistäminen. Tämän tarkoituksena oli minimoida taktisen tason moduulien välillä tapahtuva viestintä ja tekoälyn toimintojen keskittäminen yksittäisiin moduuleihin. Moduulien välisen viestinnän puuttuminen aiheutti kuitenkin ongelmia esimerkiksi tilanteissa, joissa eri moduulit olivat pyrkineet siirtämään yksiköitä samoihin ruutuihin toistensa kanssa.

Lisäksi toimintojen erillisiin moduuleihin hajauttamisen ja moduulien välisen yhteistyön minimoimisen vuoksi joidenkin toimintojen toteutuksessa tuli merkittäviä ongelmia. Esimerkiksi yksiköiden suojaamisesta vastaavalla moduulilla olisi täytynyt olla keino pyytää uusia siirtoja muilta tekoälyn moduuleilta tilanteessa, jossa yksiköitä suojaava moduuli oli päättänyt hylätä jotain resurssia koskevan siirron. Tällä hetkellä resurssi jää käyttämättä, ja mikäli yksiköitä suojaava moduuli joutuisi itse ratkaisemaan korvaavan siirron, olisi moduulin osattava kaikkien taktisen tason moduulien toimet.

ADAPTA-päättelyarkkitehtuurin tavoitteena oli parantaa tekoälyn strategista pelaamista ja siten mahdollistaa parempi suoriutuminen niissä strategisissa ongelmissa, joista Buro ja Furtak kirjoittivat artikkelissa RTS Games as Test-Bed for Real-Time AI Research. Tämä pyrittiin saavuttamaan päättelyarkkitehtuurissa siten, että tekoälyn taktisen tason moduulit suunnittelivat siirtoja vuoron alussa ja strategisen tason moduulit pystyivät näistä siirroista raken-

tamaan parhaan mahdollisen sarjan siirtoja. Vaikka ajatuksena toimintamalli on toimiva, huomattiin testauksen aikana, että se aiheutti merkittäviä ongelmia tekoälyn kyvyssä mukautua muuttuviin tilanteisiin, jotka Planet Cube -pelissä olivat mahdollisia.

Esimerkiksi tilanteessa, jossa tekoälyn ohjaama yksikkö onnistui tuhoamaan vihollisen yksikön, oli mahdollista, että usean tekoälyn ohjastaman yksikön siirrot jäivät osittain käyttämättä, mikäli yksiköt olivat aikeissa hyökätä tuhoutuneen vihollisyksikön kimppuun. Tämä oli mahdollista erityisesti tilanteissa, joissa ensimmäisten joukossa hyökkäävä yksikkö onnistui tekemään laskettua enemmän vahinkoa viholliseen ja tuhoamaan sen. Laskettua enemmän tehtävän vahingon mahdollistavat Planet Cube -pelissä kriittiset iskut (critical hit), jotka onnistuessaan tekevät moninkertaisen määrän vahinkoa kohteeseen.

Toinen vuoron suunnittelun ajankohdasta aiheutuva ongelma oli tekoälyn hidas reagoiminen pelikentältä kerättävään tietoon. Koska vuoron siirrot suunniteltiin kokonaisuudessaan vuoron alussa ennen kuin ainuttakaan siirtoa oli vielä tehty, ei tekoäly pystynyt ottamaan huomioon sellaista tietoa, joka oli kerätty kyseisen vuoron aikana, vaan sen sijaan tekoäly reagoi tähän tietoon vasta seuraavalla vuorolla. Esimerkiksi tilanteessa, jossa tekoälyn ohjaama yksikkö löytää aikaisemmin näkymättömissä olleen vihollisyksikön, muut tekoälyn yksiköt eivät voi reagoida vihollisen paljastumiseen, koska yksiköiden tekemät siirrot on jo päätetty.

Testauksen aikana yksittäisten moduulien toiminnassa ei havaittu merkittäviä ongelmia. Sen sijaan suurin osa tekoälyn ongelmista aiheutui ADAPTA-päättelyarkkitehtuurin hajanaisen rakenteen, moduulien välisen kommunikaation puutteen ja ADAPTA-päättelyarkkitehtuurin suoritusjärjestyksen seurauksena. Nämä ongelmat ovat ratkaistavissa muuttamalla päättelyarkkitehtuuria siten, että moduuleiden välillä on jokin standardoitu tapa välittää tietoa ja parantamalla tekoälyn mukautuvuutta hajauttamalla siirtojen suunnittelua useampaan vaiheeseen vuoron ajalle.

6 POHDINTA

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa tekoälyn päättelyarkkitehtuuri Planet Cube -peliprojektiin ja tämä suunnitelma toteutui. Opinnäytetyösuunnitelmassa oli kuitenkin päätetty, että suunnittelu ja toteutus ei vielä itsessään olleet riittävä saavutus opinnäytetyönä, vaan tekoälyn tulisi myös olla toimiva ja tarjota hyvä vastustaja peliä pelaaville ihmispelaajille. Mikäli tekoälyn toiminta ei olisi riittävän hyvää, tulisi tekoälyn toteutuksesta hyötyä jollain tavalla. Suunnitelmassa määriteltiin, että tilanteessa, jossa tekoälyn toiminta ei olisi riittävän laadukasta, tulisi opinnäytetyöstä käydä ilmi, millä tapaa tulevaisuudessa kehittävät seuraavat tekoälyn versiot voitaisiin tehdä paremmin.

Opinnäytetyössä toteutettu tekoäly ei loppujen lopuksi yltänyt asetettuihin laatuvaatimuksiin, joten opinnäytetyön tärkein tavoite oli kartoittaa ongelmakohdat ja jatkokehityskohteet. Merkittävin jatkokehitystä vaativa kohde tulisi olemaan tekoälyn päättelyarkkitehtuuri ja ADAPTA-päättelyarkkitehtuurin asettamista ohjenuorista irtaantuminen. ADAPTA-päättelyarkkitehtuuri voisi toimia todella hyvin peleissä, joissa kaikki strategiseen päättelyyn vaikuttava tieto on aina saatavilla tai se muuttuu ainoastaan vuorojen välissä, mutta Planet Cube -pelin kaltaisessa pelissä päättelyarkkitehtuuri oli heikoilla.

ADAPTA-päättelyarkkitehtuurissa on kuitenkin joitain hyviä puolia, jotka jatkokehityksessä tulisi ottaa huomioon. Merkittävin näistä hyvistä puolista on päättelyarkkitehtuurissa käytetty modulaarinen rakenne, joka mahdollistaa nopean ja helpon tekoälyn laajentamisen, mikäli kehittäjän asettamat vaatimukset pelin tekoälylle kasvavat. Suurimmaksi päättelyarkkitehtuurissa havaituksi ongelmaksi osoittautui tapa, jolla tekoälyn päättelyprosessi ajoitettiin. Vuoron alkuun painottuva suunnitteluvaihe, jonka aikana koko tuleva vuoro suunniteltiin kerralla valmiiksi, oli liian hidas reagoimaan pelimaailmassa ja tekoälyllä hallussa olevassa tiedossa tapahtuviin muutoksiin. Jatkokehityksessä päätöksenteko- ja suunnitteluvaihe tulisi hajauttaa koko vuoron ajalle, jolloin tekoäly pystyisi paremmin reagoimaan näihin muutoksiin.

Kaiken kaikkiaan opinnäytetyön aikana saavutetut tavoitteet ja toteutuksen aikana kerätyn tiedon pohjalta voidaan todeta opinnäytetyön onnistuneen. Vaikka tekoäly ei ole vielä riittävän laadukas pelin julkaisua varten, opittiin tekoälyn toteutuksen testauksen aikana sellaista tietoa, joka tulee mahdollistamaan julkaisukelpoisen tekoälyn kehityksen tulevaisuudessa.

LÄHTEET

Bergsma, M. & Spronck, P. 2008. Adaptive Spatial Reasoning for Turn-based Strategy Games.

Ahlquist, J. & Novak, J. 2008. Game Development Essentials: Game Artificial Intelligence. Thomson Delmar Learning.

Schwab, B. 2004. AI Game Engine Programming, Hingham: Charles River Media Inc.

Woodcock, S. 2002. Recognizing Strategic Dispositions: Engaging the Enemy. Teoksessa AI Game Programming Wisdom, Hingham: Charles River Media Inc, 221 – 232.

Sweetser, P. 2004. Strategic Decision-Making with Neural Networks and Influence Maps. Teoksessa AI Game Programming Wisdom 2, Hingham: Charles River Media Inc, 439 – 446.

Wetzel, B. 2008. The Engagement Decision. Teoksessa AI Game Programming Wisdom 4, Boston: Course Technology, 443 – 454.

Champandard, A. 2011. The Mechanics of Influence Mapping: Representation, Algorithm & Parameters. Web-dokumentti. Saatavilla: http://aigamedev.com/open/tutorial/influence-map-mechanics/ (Luettu 4.3.2013).

Buro, M. & Furtak, T. 2003. RTS Games as Test-Bed for Real-Time AI Research.

TEKOÄLYN LUOKKAKAAVIO

AdaptaModule
- m_managingAI : AdaptaAI
+ <<constructor>> AdaptaModule()
+ SetManagingAI(managingAI:AdaptaAI) : void
AdaptaStrategyModule
+ <<constructor>> AdaptaStrategyModule()
+ Run(data:object) : object
IntelligenceModule
- m_enemyUnits : List<uint>
- m_enemyAttackAreas : Dictionary<uint, List<uint>>
- m_threatMap : InfluenceMap
+ <<constructor>> IntelligenceModule()
+ Run(data:object) : object
AssetAllocationModule
+ <<constructor>> AssetAllocationModule()
+ Run(data:object) : object
UtilityManagementModule
+ <<constructor>> UtilityManagementModule()
+ Run(data:object) : object
ProtectionModule
+ <<constructor>> ProtectionModule()
+ Run(data:object) : object
ActionOrderingModule
+ <<constructor>> ActionOrderingModule()
+ Run(data:object) : object
AdaptaTacticalModule
+ <<constructor>> AdaptaTacticalModule()
+ GenerateBids(List<AdaptaResource> availableResouces) : List<AdaptaBid>
+ GenerateActions(List<AdaptaResource> allocatedResources) : List<AdaptaActionSet
UnitMovementModule
+ <<constructor>> UnitMovementModule()
+ GenerateBids(List<AdaptaResource> availableResouces) : List<AdaptaBid>
+ GenerateActions(List<AdaptaResource> allocatedResources) : List<AdaptaActionSet>
AttackModule
+ <<constructor>> AttackModule()
+ GenerateBids(List<AdaptaResource> availableResouces) : List<AdaptaBid>
+ GenerateActions(List<AdaptaResource> allocatedResources) : List<AdaptaActionSet>
ReconnaissanceModule
+ <<constructor>> ReconnaissanceModule()
+ GenerateBids(List<AdaptaResource> availableResouces) : List<AdaptaBid>
+ GenerateActions(List<AdaptaResource> allocatedResources) : List<AdaptaActionSet>