LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DAUDZAĢENTU SISTĒMAS UN TO VIZUALIZĀCIJA

BAKALAURA DARBS

Autors: **Hardijs Ķirsis**
Studenta apliecības Nr.: hk14004
Darba vadītājs: Dr. dat. Guntis Arnicāns

RĪGA 2018

ANOTĀCIJA

Bakalaura darba mērķis ir veikt analīzi, meklēt optimālu risinājumu, lai ar datora palīdzību spētu attēlot dažādas daudzaģentu sistēmas. Lai to panāktu, tika pētītas, ievākti fakti un veikti secinājumi par to kas īsti ir daudzaģentu sistēma, kādi tām ir pielietojumi, kā arī to raksturīgākās sastāvdaļas. Problēma - ar viena rīka palīdzību attēlot vairākus simulacijas scenārijus, nezinot to jēgu un nepieciešamo iznākumu. Iegūtie dati tika sakopoti, ar kuru palīdzību, autors aprakstīja iespējamo attēlošanas risinājuma izveidi un izveidoja prototipu. Izstrādes gaitā autors secināja, ka tā nav tik triviāla problēma, kā sākumā likās.

ATSLĒGVĀRDI: Daudzaģentu sistēmas, simulācijas, vizualizācija, tīmekļa tehnoloģijas

ABSTRACT

MULTI-AGENT SYSTEMS AND THEIR VISUALIZATION

This thesis is written in order to perform analysis, look for an optimal solution to be able to display various multi-agent systems with help of a computer. To accomplish such a task, investigation about multi-agent systems had to be made, what are they used for and what exactly are they, what kind of characteristics they possess. Problem – how would it be possible to visualize a wide variety of simulations not knowing its meaning and the desired outcome. Upon gathering necessary information, author used them to describe his pruposed solution and created a prototype. During the development, the author concluded that this is not as trivial problem as it first appeared.

KEYWORDS: Multi-agent systems, simulations, visualization, web technologies

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

Apzīmējums	Skaidrojums
JavaScript	Programmēšanas valoda, kas ir ļoti izplatīta tīmekļu veidošanā
Canvas	HTML elements, kurš paradzēts dažādu grafisko elementu attēlošanā
NetLogo	Daudzaģentu sistēmu modelēšanas vide, programmēšanas valoda
Repast Symphony	Daudzaģentu sistēmu modelēšanas vide
Groovy	Programmēšans valoda
ReLogo	Programmēšans valoda
OBJ	Datne, kura satur 3D objekta ģeometriju
JSON	Datu apmaiņas standarts

IEVADS

Daudzagentu sistēma ir sistēma, kas sastāv no vairākiem indivīdiem jeb aģentiem, kuri atrodas kādā konkrētā vidē, aģenti varētu būt roboti, cilvēki, tai skaitā programatūra, kura ir spējīga pieņemt lēmumus un patstāvīgi rīkoties vai arī pavisam kaut kas cits, kas spēj izrādīt autoritāti, personību, intelektu utml.

Datorzinātnēs daudzaģentu sistēmu var uzskatīt par datorizētu sistēmu ar kuras palīdzību tiek veiktas dažādas simulācijas vai citas līdzīgas darbības, kur nepieciešams kaut ko attēlot vai radīt. Šāda veida simulācijas palīdz pētīt grupveida sabiedrības, kuras sastāv no vairākiem indivīdiem, kuri bieži vien seko kādiem noteikumiem.

Lai šīs sistēmas būtu vieglāk pētīt, tās nepieciešams padarīt cilvēkam viegli uztveramas, kas rada nepieciešamību tās vizuāli attēlot, kas savukārt rada izaicinājumu, it īpaši tad, ja tiek attēlota sarežģīta daudzaģentu sistēma. Kvalitatīvas attēlošanas pamatprincips - lietotājam būtu viegli jāuztver modeļa galvenā doma, nevajag izdaiļot to[1].

Šādiem nolūkiem ir radīti rīki, kur problēmas modelētājs varētu šo problēmu attēlot un pētīt, neskatoties uz to, ka pamatprincips šiem rīkiem ir līdzīgs, strādāt ar katru ir mazliet savādāk, nepieciešamas dažādas programmēšanas zināšanas, kā arī izvēlēties īsto rīku nemaz nav tik acīmredzami, katram ir savas pozitīvās un negatīvās puses. Būtu ļoti noderīgi izveidot risinājumu, kurš būtu neatkarīgs no visiem šiem rīkiem, nepieciešamības gadījumā, lietotājs varētu izstrādāt arī savu daudzaģentu sistēmu simulāciju savā iecienītajā programmēšanas valodā, nebūtu spiest izvēlēties kādu esošu rīku, programmēšanas valodu, lai to attēlotu.

Šī bakalura darba mērķis ir, izpētot vairākas daudzaģentu sistēmas un to raksturīgākās īpašības, uzbūvi, izveidot abstraktu daudzaģentu sistēmu, kas raksturotu šāda veida sistēmu svarīgākās komponentes un īpašības, kuras būtu būtiski saprast daudzaģentu sistēmu izstrādes un to attēlošanas procesā. Izveidoto daudzaģentu sistēmas abstrakciju autors izmantoja, lai nonāktu pie risinājuma ar kura palīdzību būtu iespējams attēlot plašu klāstu daudzaģentu sistēmu.

Mērķa sasniegšanai tika izvirzīti šādi uzdevumi:

- Iepazīties ar daudzaģentu sistēmām, to pielietojumiem
- Analizēt dažādu jau izveidotu daudzaģentu sistēmu
- Izveidot abstraktu daudzaģentu sistēmu
- Iepazīties ar dažādām grafisko elementu attēlošans iespējām tīmeklī
- Pielietojot izveidoto abstrakto daudzaģentu sistēmas aprakstu, meklēt vienotu attēlošanas risinājumu, izveidot tā prototipu

Darbs sastāv no 5 sadaļām. *Daudzaģentu sistēma* iepazīstina ar šāda veida sistēmām, pielietojumiem. *Daudzaģentu sistēmu esošo rīku apskats* sniedz ieskatu par šādu sistēmu izstrādes izmantotajiem rīkiem, to iespējām un problēmām. *Daudzaģentu sistēmu attēlošana tīmeklī* tiek izvirzīti kritēriji, kurus būtu ieteicams ņemt vērā, attēlojot daudzaģentu sistēmas tīmeklī, attiecīgi izvēlēta tehnoloģija. *Daudzaģentu sistēmu analīze* sadaļā autors pētīja dažādas daudzaģentu sistēmas un to realizācijas, iegūstot raksturīgākās šo sistēmu īpašības, kuras tika izmantotas prototipa izstrādē. *Rīka realizācija* sadaļā tiek minētas izvēlētās tehnoloģijas, rīka uzstādījumi, aprasktīta struktūra ar kuras palīdzību var attēlot plašu klāstu daudzaģentu sistēmu.

1. DAUDZAĢENTU SISTĒMA

Datorzinātnēs daudzaģentu sistēmu var uzskatīt par datorizētu sistēmu ar kuras palīdzību tiek veiktas dažādas simulācijas vai citas līdzīgas darbības, kur nepieciešams kaut ko attēlot vai radīt. Šāda veida simulācijas palīdz pētīt grupveida sabiedrības, kuras sastāv no vairākiem indivīdiem jeb aģentiem.

Aģentu bāzētu modelēšanu ieteicams lietot, ja modelētajai problēmai piemīt kāda no šīm īpašībām [2, 3, 4]:

- Problēmu iespējams attēlot ar vairāku indivīdu palīdzību
- Indivīdiem jāspēj izrādīt rīcību, uzvedību
- Indivīdiem jāspēj mācīties, pielāgoties, mainīties
- Indivīdiem jāspēj veidot dažādus kontaktus ar pārējiem indivīdiem vai apkārtējo vidi
- Problēmas būtiska daļa ir komūnu, organizācijas, baru modelēšana, pētīšana
- Indivīdiem jāatrodas kādā diskrētā telpā

1.1. Aģents

Aģents ir kādas sistēmas neatkarīga komponente, kura uzvedība var būt pavisam vienkārša, proti, aprakstīta ar loģiskām izvēlēm - kondicionāļiem vai stipri sarežģītāka, pārklājoties ar kognīcijas un mākslīgā intelekta pētijumiem. Neskatoties uz to, aģenti bieži vien seko konkrētiem noteikumiem un ievēro dažādus protokolus, kuri apraksta aģentu savstarpējo komunikāciju vai kādu citu vēlamo rīcību, piemēram, pārvietošanos baros, citu aģentu atpazīšana, orientēšanās apkārtējā vidē. Šādā veidā iespējams radīt nepieciešamos apstākļus kādas problēmas pētīšanā vai risināšanā. Aģentiem var piemist ļoti sareģģītas īpašības, taču tas nav priekšnosacījums, jo nav izveidota universāla definīcija, kas spētu detalizēti raksturot šo termu [4, 5].

1.2. Vide

Daudzaģentu sistēmas neatņemama sastāvdaļa ir tās vide, kurā atrodas aģenti, tā kā šāda veida sistēmas atrodas virtuālajā vidē, aģentu atrašanās vieta un tās raksturojums ir cilvēka iztēles robežās. Vide var drastiski iespaidot aģentus, atkarībā no vēlemā rezultāta, tā var palīdzēt aģentiem vai tieši otrādi - kaitēt tiem, taču tā var būt arī neitrāla un kalpot tikai kā fons.

1.3. Daudzaģentu sistēmu pielietojumi

Kā jau iepriekš tika minēts - liela nozīme daudzaģentu sistēmām ir to spēja simulēt kādu problemātisku scenāriju. Sistēma, kura tiek pētīta ar simulāciju jeb daudzaģentu sistēmu

palīdzību bieži vien ir sarežģīta, neintuitīva. Šāda veida pētēmajām problēmām var nebūt izrēķināms vai paredzems iznākums, tādēļ šādas problēmas bieži tiek pētītas mainot simulācijas parametrus un gaidot rezultātu pēc kura iespējams izdarīt secinājumus un varbūt dziļāk izprast problēmu.

Daudzaģentu sistēmu pielietojums neaprobežojas ar simulācijām vai kādas sarežģītas problēmas pētīšanu, mūsdienās daudz lietas tiek automatizētas ar datoru palīdzūbu, ja šis darbs tiek sadalīts uz vairākiem indivīdiem, kuri vairāk vai mazāk virzās uz vienu kopīgu mērķi vai kādā citā veidā izrāda savu klātesamību vai piederību kopienai, tad šo sistēmu arī varam saukt par daudzaģentu sistēmu. Darba autors 1.1. tabulā izvirzītās sfēras galvenokārt iedevsmojās no *NetLogo.*

1.1. tabula

Daudzaģentu sistēmu pielietojumi

Sfēra	**Pielietojums**
Māksla	Aģentiem mainot vizuālo izskatu un pozīciju telpā, iespējams attēlot kādu vēlamu ainu vai ģenerēt mākslas darbus ar nejaušības principu
Bioloģija	Aģentus pielīdzinot dzīvām būtnēm, iespējams modelēt kādu sistēmu no dabas, piemēram, skudru pūzni, tādējādi gūstot pastiprinātu izpratni par pētēmo sistēmu. Šajā sfērā aģenti neaprobežojas ar dzīvām būtnēm, tie var būt dažādi mikroorganismi kā, piemēram, vīrusi
Ķīmija	aģentus pielīdzinot dažādām ķīmiskām vielām vai molekulām, iespējams novērot dažādu ķīmisko reakciju iznākumus vai pareģojumus.
Fizika	Aģentus pielīdzinot fiziskiem objektiem, kuri pakļaujas kādiem fizikas likumiem, iespējams modelēt dažādas situācijas, kuras reālajā pasaulē nav iespējams vai sarežģīti realizēt
Datorzinātne	Dažādu algoritmu pētīšana, arhitektūru un datortīklu attēlošana, piemēram, tūringa mašīnas vizualizācija, kā arī dažāda veida automatizācija un šo sistēmu novērošana, piemēram, sistēmu aizsardzība pret datorvīrusa palaišanu tīklā
Ģeogrāfija	Dabas stihiju, katastrofu, reljevu attēlošana, cilvēka ietekme uz vidi, laikapstākļa maiņas, piemēram, globālās sasilšanas simulācija

Matemātika	Dažādu funkciju vizualizācija, piemēram, nejaušības, statistikas pētīšana, kā arī dažādu sakarību meklēšanā, minēšana.

Uzskaitītās sfēras un to pielietojumi tajās, kas redzami 1.1. tabulā ir tikai neliels ieskats, kas nav jāuztver par pilnīgu.

2. DAUDZAĢENTU SISTĒMU ESOŠO RĪKU APSKATS

Tīmeklī ir sastopami visdažādākie datu vizualizācijas risinājumi, kurus varbūt varētu izmantot pat daudzaģentu sistēmu attēlošanā, taču darba autors izvēlējās tikai tos risinājumus, kuri ir veidoti tieši priekš daudzaģentu sistēmu veidošanas un attēlošanas, neatkarīgi no tā vai ir nepieciešamas programmēšanas zināšanas šo rīku izmantošanā vai nē.

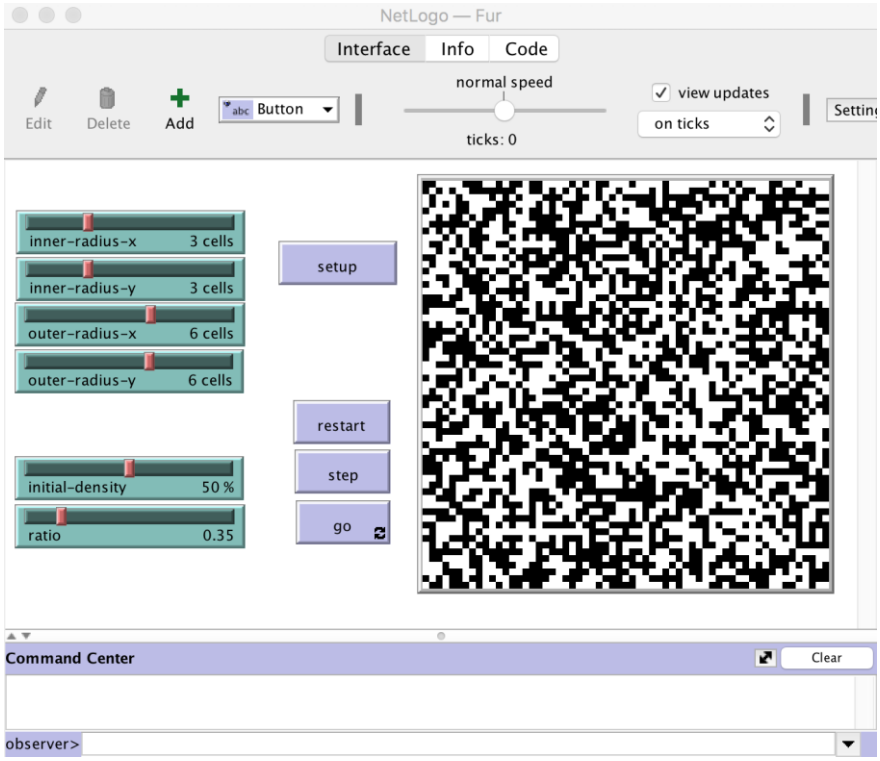
Izvēlētie rīki tika salīdzināti pēc šādiem kritērijiem:

- **Attēlošanas izstrādes iespējas** - objektu rezprezentēšana ar modelēšanas palīdzību, versiju kontrole, iespēja sastrādāties kolektīvā
- **Attēlotās simulācijas lietojamība** - skatu maiņa, attēlotās simulācijas ātruma maiņa, atkārtota iespēja noskatīties simulāciju, 2D un 3D telpu atbalsts
- **Simulācijas datu eksportēšana** - vai iespējāms eksportēt simulācijas datus, lai tos varētu izmantot kāda cita programmatūra; datņu formāti
- **Vides pārklājums** - kādās vidēs iespējams attēlot simulāciju

2.1. Netlogo

Netlogo ir atvērtā pirmkoda programatūra, kura nodrošina konfigurējamu modelēšanas vidi priekš dažādu simulāciju veidošanas. To izstrādāja Uri Wilensky 1999. gadā, taču izstrāde pie šī rīka nav beigusies. *Netlogo* izstrādātāji uzskata, ka šis rīks ļoti labi iederas dažādu sarežģītu sistēmu modelēšanā, kuras laika gaitā mainās un attīstās, taču gana vienkāršs, lai to spētu izmantot studenti un skolotāji, kā arī pietiekami aprīkots, lai to izmantotu dažādu nozaru profesionāļi. [6]

Netlogo piedāvā vidi ar kuras palīdzību ir iespējams mainīt dažādus simulācijas parametrus un novērot to ietekmi uz modeli, tajā pašā laikā sniedzot dažādu grafiku attēlošanu, kas iespējams var sniegt dziļāku izpratni par simulāciju (skat. 2.1.att). Izstrādātājam ir iespēja pievienot dažādas pogas un ievades laukus, kuri paradzēti simulācijas parametru un vērtību iestatīšanai.

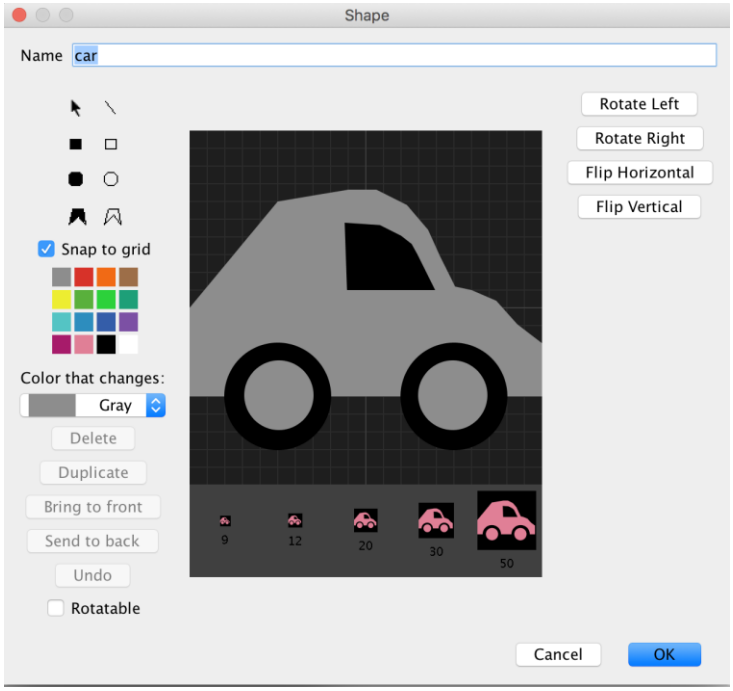


2.1. att. ***Netlogo*** **rīka saskarne**

Kā arī šis rīks piedāvā plašu klāstu ar jau izstrādātām sistēmām, kuras lietotājs var pētīt, kā arī ir tiesīgs labot pēc saviem uzskatiem, ja nepieciešams mainīt modeļa jēgu.

Netlogo novērtējums pēc kritērijiem:

- **Attēlošanas izstrādes iespējas** - diemžēl nav iespējams importēt projektā nekādus standartizētus 2D vai 3D modeļa failus, kā, piemēram, *OBJ* formātu vai citus, taču *Netlogo* piedāvā plašu klāstu ar jau izstrādātiem modeļiem, kā arī iebūvētu rīku modeļu izveidē, bet tas ir neadekvāts un ļoti novecojis (skat. 2.2. att); Versiju kontrole rīkā nav iebūvēta, taču iespējams eksportēt projektu un to ievietot kādā pirmkoda glabātuvē, jo eksportētais fails ir teksta formātā, kurš satur simulācijas loģiku un vizuālos elementus. Ir iespējams arī eksportēt projektu pa daļām, taču jāatdzīst, ka nav iespējams eksportēt vizuālos elementus atsevišķi, kas piespiež lietotājus eksportēt projektu vienā veselā failā; Ir iespējams importēt projektu un tādā veidā kolektīvi izstrādāt sistēmu, taču jāatdzīst, ka tiek eksportēts viens fails, kas noteikti apgrūtinātu sistēmas izstrādi pie lielākiem projektiem, kā arī piesārņotu kodu glabātuvi ar 2D un 3D modeļa informāciju;



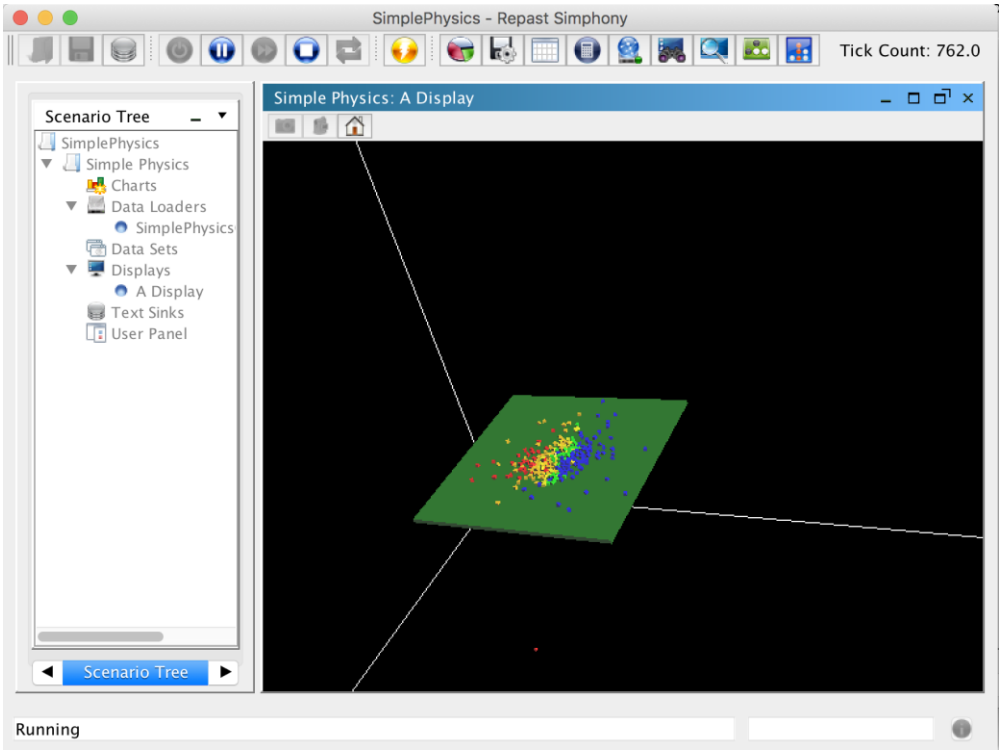
2.2. att. ***Netlogo*** **modeļu izstrādes rīks**

- **Attēlotās simulācijas lietojamība** - plaša skatu maiņa (rotācija ap orbītu kādā konkrētā punktā, pietuvināšana un attālināšana, neierobežota kustība pa simulācijas telpu, netraucējot tā klātesošos); Iespējams mainīt simulācijas ātrumu un pilnībā apturēt to; Netika atrasta iespēja atkārtoti noskatīties tikko nosimulētās darbības, kā tikai eksportēt ekrānšāviņus; Rīks atbalsta gan 2D gan 3D telpas;
- **Simulācijas datu eksportēšana** - Rīkā ir iebūvēta funkcionalitāte, kas nodrošina dažādu objektu eksportēšanu, galvenokārt - aģentu un vides informācija; csv,txt datnes formāti
- **Vides pārklājums** - plaši pieejams rīks (*Mac*, *Windows*, *Linux* un tīmekļa vide), taču tika konstatēts, ka tīmekļa vide satur tikai mazu daļu no pārējo platformu iespējām. Kā vislielākos trūkumus jāatdzīst nespēju izveidot daudzaģentu sistēmas, bet tikai to attēlošanu, kas arī ir ļoti nepilnīga, jo neatbalsta kustību pa 3D telpu [7].

2.2. Repast Simphony

Repast ir atvērtā pirmkoda ietvars, kura galvenais mērķis, līdzīgi kā *Netlogo*, ir simulēt dažādas aģentu bāzētus scenārijus, kurus varētu pētīt un attēlot. Šo rīku izstrādā jau vairāk nekā 15 gadus un izstrāde vēl nav gala [8]. Sākumā to izstrādāja David Sallach, taču tagad pie tā darbojas vairāku cilvēku komanda [9]. Šis rīks ir izstrādāts Java programmēšanas valodā, kura izstrādātājiem ļauj programmēt ne tikai *Java* programmēšansa valodā, bet *Groovy*, kā arī mazāk pieredzējušiem izstrādātājiem dod iespēju izmantot draudzīgāku *ReLogo* programēšanas valodu [10].

Līdzīgi kā *Netlogo, Repast Simphony* ietvara lietotājiem arī ir iespēja pievienot dažādas pogas un ievades laukus, kuri paradzēti simulācijas parametru un vērtību iestatīšana, ar kuriem manipulējot, var novērot to ietekmi uz modeli (skat. 2.3.att.).



2.3. att. **Repast Simphony simulācijas veikšana**

Repast novērtējums pēc kritērijiem:

- **Attēlošanas izstrādes iespējas** - šis rīks nepiedāvā 2D vai 3D modeļu izstrādi, taču piedāvā lietotājam draudzīgu Java klašu kopumu ar kura palīdzību ir iespējams veidot dažādus objektus, kā arī ielādēt bildes un standartizētus modeļa failus kā, piemēram, *OBJ* un *3DS* [6]; Projekta struktūru izstrādātājs var veidot sev izdevīgā veidā, kā arī projekts sastāv no vairākiem teksta failiem, kuru iespējams atvērt ar kādu *Java IDE*, tā rezultātā varam secināt, ka versiju kontroles iespējas ir plašas; Projektu iespējams importēt un eksportēt, izmantojot kādu *Java IDE*, tādējādi šo rīku ir ērti izmantot arī izstrādātāju komandām.
- **Attēlotās simulācijas lietojamība** - plaša skatu maiņa (rotācija ap orbītu kādā konkrētā punktā, pietuvināšana un attālināšana, neierobežota kustība pa simulācijas telpu, netraucējot tā klātesošos); Iespējams mainīt simulācijas ātrumu un pilnībā apturēt to; Pastāv iespēja simulācijas laikā eksportēt ekrānšāviņus, kā arī video, taču tas neļauj lietotājam pilvērtīgi aplūkot vizualizāciju pienācīgi, jo nav iespēja manipulēt ar skatu. Simulācijas laikā arī nav iespējams atgriesties iepriekšējajā solī; Rīks atbalsta gan 2D gan 3D telpas;

- **Simulācijas datu eksportēšana** - rīkā ir iebūvēta funkcionalitāte - *freeze dry*, kas nodrošina dažādu objektu eksportēšanu, galvenokārt - aģentu un vides informācija; xml datnes formats [11].
- **Vides pārklājums** - plaši pieejams rīks (*Mac, Windows, Linux*, taču nav iespējams palaist tīmeklī) [9].

2.3. Apkopojums

Aplūkotie rīki - *Netlogo* un *Repast Symphony,* pilda nepieciešamās funkcijas, kuras izstrādātājs varētu sagaidīt no šāda veida rīkiem - iespējams izstrādāt gan sarežģītas gan vienkāršas daudzaģentu sistēmas krietni īsākā laika periodā nekā, ja izstrādātājam pašam būtu jārealizē simulācijas loģika un telpas attēlošana. Kopumā abi risinājumi šķiet līdzvērtīgi, taču *Netlogo* ir vieglāk lietojams, bet ierobežojošāks. Aplūkojot šos rīkus gan jāatdzīst, ka abi šie risinājumi šķiet vizuāli novecojuši, neskatoties uz to, ka abu rīku izstrāde vēl nav galā. Tas iespējams ir viens no iemesliem kādēļ darba autors novēroja dažu svarīgu funkciju iztrūkumu vai nepilnību abos risinājumos:

- Nav iespējams pienācīgi attēlot daudzaģentu sistēmas tīmeklī, jo netiek atbalstīti 3D modeļi un telpa
- Nav iespējams attēlot tikko simulētās darbības, neizmantojot iebūvēto ekrānšāviņu un video ierakstīšanas funkcionalitāti, kas krietni ierobežo simulācijas pētīšanu, jo nav iespējams rotēt, kustēties pa telpu utml
- Lai attēlotu kādu simulāciju nepieciešams lietot kādu no abu rīku atļautājām programmēšanas valodām
- Simulācija un tās attēlošana notiek reālā laikā un nav iespējams to mainīt, kas noteikti nav vislabākais risinājums pie ļoti lielām vai sarežģītām sistēmām, jo vienlaicīgi tiek veikta simulācija un tās attēlošana, kas var nopietni ietekmēt attēlošanas kvalitāti

3. DAUDZAĢENTU SISTĒMU ATTĒLOŠANA TĪMEKLĪ

Time vietnes dati norāda to, ka tīmekļa tehnoloģiju lietotāju skaits strauji aug, 2000. gadā tie bija 400 miljonu lietotāju, taču 2015. gadā to skaits sasniedzis jau 3.2 miljardus, kas nozīmē to, ka 2015. gadā gandrīz katrs otrais pasaules iedzīvotājs izmanto tīmekļa tehnoloģijas. Nav pamats domāt, ka nākotnē šis skaits nepalielināsies, jo mazāk attīstītākās valstīs lietotāju pārklājums pagaidām ir salīdzinoši mazs. Dati vēsta arī to, ka liela daļa no šī pienesuma nāk tieši no mobīlajām ierīcēm [12, 13].

Tākā daudzaģentu sistēmas mēdz būt apjomīgas un sarežģītas, darba autors secināja, ka vislietderīgāk būtu pētīt tīmekļa bāzētus risinājumus, kuri neaprobežojas tikai ar *CPU* resursa izmantošanu, bet spēj izmantot arī *GPU,* jo *GPU* ir daudz spējīgāks dažādu grafisko objektu attēlošanā [14]. Par otru kritēriju tika izvirzīts 2D un 3D modeļu un telpas atbalsts.

Darbā tika aplūkotas *WebGL* un *Java Applet* tehnoloģijas, jo abas tehnoloģijas apmierināja izvirzītos uzstādījumus.

3.1. WebGL

Šo tehnoloģiju sākotnēji ieviesa bezpeļņas organizācija *Khronos Group* 2011. gadā ar mērķi izveidot 3D grafisko elementu attēlošanas standartu, kas atbalstītu plašu klāstu ierīču, taču tehnoloģijas ideja radās jau 2006. gadā, kad Vladimir Vukićević veica eksperimentus ar HTML Canvas elementu.

WebGL ir JavaScript lietojumprogrammu saskarne, kura nodrošina zema līmeņa sistēmas izsaukumus ar *OpenGL ES* palīdzību, tādējādi tā spēj izmantot ierīces *GPU*, lai ātrāk veiktu nepieciešamās kalkulācijas un tās rezultātus spēj paradīt tīmekļa vietnē izmantojot *HTML5 Canvas* elementu, neliekot lietotājam izmantot papildus programmatūru [15]. Tādā veidā tā ļoti viegli integrējās tīmekļa vietnē.

3.1.1. WebGL integrācija tīmekļa vietnē

Kā jau iepriekš minēts, *WebGL* izmanto *HTML Canvas* elementu. *Canvas* elements kalpo kā konteineris, kurā iespējams attēlot dažādus grafiskus elementus ar JavaScript programmēšanas valodas palīdzību [16]. 1.pielikumā var aplūkot *HTML* pirmkoda piemēru šī elementa pievienošanai tīmekļa vietnē. Lai spētu izmantot *WebGL*, nepieciešams to izveidot ar *Canvas* elementa palīdzību, *HTML* pirmkoda paraugs apskatāms 1. pielikumā.

Pēc WebGL integrācijas tīmekļa vietnē, to iespējams izmantot. Tīmeklī pieejami daudz ietvaru, kuri atvieglo *WebGL* lietojumprogrammatūras saskarni, tādējādi uzlabojot tā lietojamību. Šos ietvarus galvenokārt izmanto dažādu datu attēlošanā, videospēļu un citu interaktīvu tīmekļu tehnoloģiju veidošanā [17, 18].

3.2. Java applet

Java applet ir viens no Javas lietotņu paveidiem, kurš tiek kompilēts uz Java bitkodu. Šo tehnoloģiju ieviesa kompānija *Sun Microsystem*s 1995. gadā, kuras mērķis bija izvietot Java bitkoda kompilētās lietotnes tīmekļa vidē, kas tika realizēts ar *JVM* palīdzību.

Neskatoties uz to, ka *Java applet* iespējams ievietot tīmeklī, tās izpilde nenotiek uz tīmekļa pārlūkprogrammām, bet gan uz *JVM*, to iespējams panākt ar Java pārlūkprogrammu spraudņa palīdzību. Šāds tehnoloģijas risinājums Java applet lietotnei nodrošina vidi, kur tā spēj izmantot ierīces *GPU*, lai ātrāk veiktu nepieciešamās kalkulācijas un tās rezultātus spēj paradīt tīmekļa vietnē izmantojot *HTML* applet elementu, kā arī *JavaScript* [15, 19].

3.2.1. Java applet integrācija tīmekļa vietnē

Ņemot vērā tehnoloģijas izstrādātāja ieteikumus neizmantot *HTML* applet elementu lietotnes integrācijai tīmekļa vietnē, jo tas var apdraudēt lietotāju drošību, integrācijai nepieciešams [20]:

- Java applet lietotne, kuru nepieciešams novietot uz tīmekļa servera
- JNLP datne, kuras paraugs aplūkojams 2.pielikumā
- Ar JavaScript palīdzību *HTML* datnē jānorāda uz *JNLP* datni, pirmkoda paraugs aplūkojams 2.pielikumā

3.3. Apkopojums

Ņemot vērā nodaļā minētās tīmekļu tehnoloģiju lietotāju tendences, darba autors secināja, ka daudzaģentu sistēmas attēlošanā *WebGL* tehnoloģija būtu piemērotāka, jo nav nepieciešama papildus programmatūra, kas radītu problēmas mobilo lietotāju vidū, tādējādi vairāk lietotāju spētu to aplūkot.

4. DAUDZAĢENTU SISTĒMU ANALĪZE

Lai izveidotu daudzaģentu sistēmas abstrakciju, tika analizētas vairākas daudzaģentu sistēmas un to vizalizācijas, tādā veidā iegūstot to raksturīgākās īpašības un atribūtus. Darba autors izvēlējās pētīt *NetLogo* piedāvātās sistēmas, jo šis rīks tiek plaši izmantots stundentu un dažādu nozaru speciālistu vidū. Katram indivīdam ir iespēja izveidot savu daudzaģentu sistēmu un publicēt to *NetLogo*. Šīs sistēmas tiek analizētas, atkarībā no kvalitātes un pienesuma - tās vai nu tiek pievienotas vai noraidītas. Tādējādi *NetLogo* ir ieguvis kvalitatīvas un unikālas daudzaģentu sistēmas, kuras darba autors izmantos, lai izveidotu abstraku daudzaģentu sistēmas aprakstu.

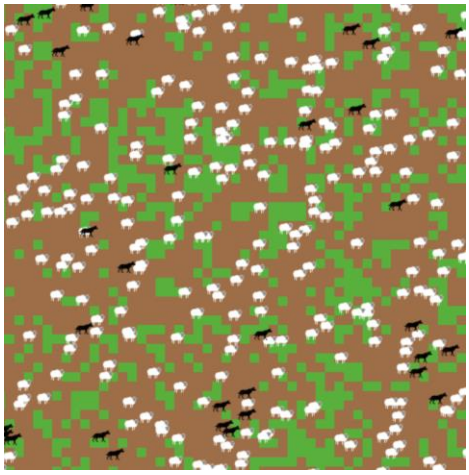
Lai nekļūdīgi novērotu šīs iezīmes, ir nepieciešams aplūkot pilnīgi visas daudzaģentu sistēmas, taču tas nav praktiski izdarāms, tadēļ nepieciešams tās klasificēt. Klasifikācijai tika izmantota 1.1 tabula.

Klasifikācija:

- Māksla
- Bioloģija
- Ķīmija
- Fizika
- Datorzinātne
- Ģeogtāfija
- Matemātika

Darba autors izvēlējās šādu klasifikāciju, lai pēc iespējas vairāk samazinātu nepieciešamo daudzaģentu sistēmu skaitu, kurām būtu jāveic analīze, taču nezaudējot sfēras pārklājumu, kurās šīs sistēmas tiek izmantotas.

4.1. Aitu un vilku sistēma



4.1. att. **NetLogo aitu un vilku simulācijas attēlojums 2D telpā**

4.1. att. attēlā redzamā sistēma sastāv no divu aģentu veidiem – aita un vilks. Aitas ēd zāli, vilki ķer aitas, abi aģenti vairojās un mēģina izdzīvot.

4.1. tabulā tiek apkopotas novērotās simulācijas vides iezīmes.

4.1. tabula

Aitu un vilku sistēmas vides iezīmes

Vides iezīme	**Iezīmes apraksts, vērtība**
Fragmentēta	Redzamais laukums ir sadalīts vienādos, mazākos apgabalos
Fragmenta forma	Kvadrāts
Krāsa	Mainīga, brūna vai zaļa
Fragmenta izmērs	Konstants, izmērāms
Fragmenta pozīcija	Konstanta
Fragmenta rotācija	Nav
Fragments satur vērtību	● Zāle ir ataugusi ● Zāles ataugšanas laiks
Neatkarība	Katrs fragments ir unikāls un nesaistīts ar citu
Fragmentu Identificējamība	Iespējama, bloki nepārklājas, taču saplūst kopā, ja to krāsas ir vienādas, kas dod iespaidu, ka vide ir viens liels veselums

4.2. tabulā tiek apkopotas simulācijas novērotās aģentu iezīmes.

4.2. tabula

Aitu un vilku sistēmas aģentu iezīmes

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Aita vai vilks
Krāsa	Aģents var sastāvēt no vairākām krāsām - melna, balta, pelēka
Izmērs	Konstants, izmērāms
Pozīcija	Aģenta pozīcija ir mainīga, tas spēj pārvietoties ļoti vienmērīgi,

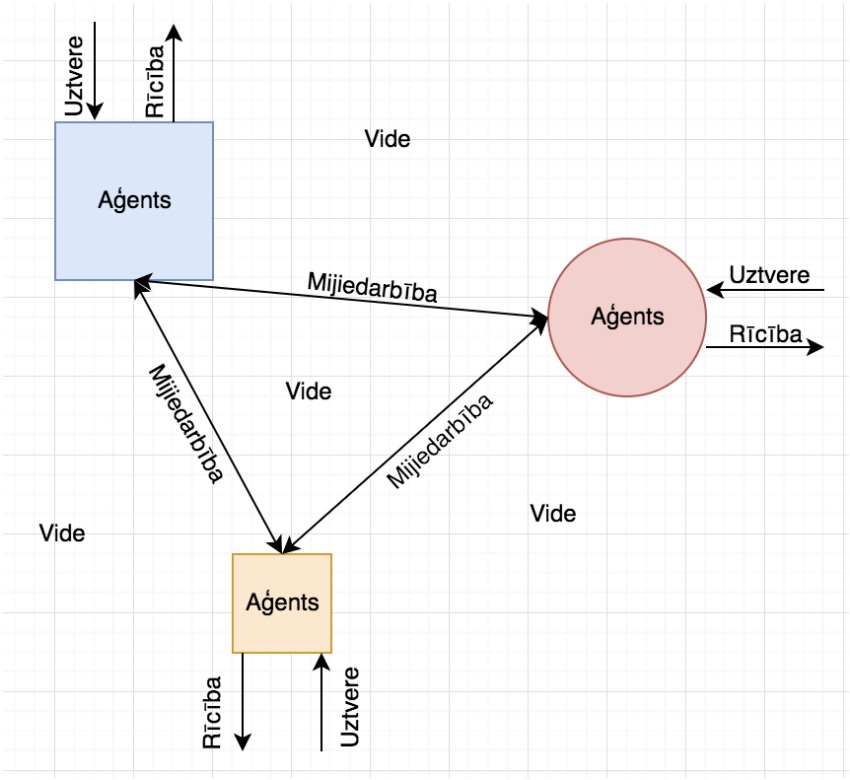
	kustība netiek limitēta tikai pa vides fragmentiem - tas var atrasties starp tiem, kā arī pamest vides redzamo daļu. Aģentam var būt tikai viena pozīcija.
Rotācija	Mainīga, visos virzienos
Atstāj iespaidu uz vidi	Maina videi krāsu - no zaļas uz brūnu
Piederība grupai	Aģents ieņem lomu - vilks vai aita, attiecīgi rīkojās
Kontaktējas, ietekmē citus aģentus	Abi aģenti vairojās, vilki nogalina aitas
Identificējamība	Iespējama, taču aģents var aizsekt otru aģentu
Satur vērtību	Uztura nepieciešamības laika atskaite

3. pielikumā var apskatīt pārējās aplūkotās daudzaģentu sistēmas, kur tiek analizētas vizuālās īpašības dažādām simulācijām.

4.2. Abstraka daudzaģentu sistēma

Veicot dažādu daudzaģentu sistēmu analīzi, vadoties pēc iegūtajiem datiem un izmantoto literatūru, darba autors nonāca pie šo sistēmu kopīgajām iezīmēt, kuras varētu uzskatīt par šo sistēmu pamatstruktūru. Iegūtās iezīmes un īpašības ir noderīgi zināt vairāku iemeslu dēļ:

- Ar kādām īpašībām iespējams operēt, lai efektīvāk attēlotu nepieciešamo daudzaģentu sistēmu
- Kuras īpašības ir nepieciešams nodrošināt, lai spētu attēlot daudzaģentu sistēmas, neatkarīgi no modeļa jēgas



4.2. att. **Daudzaģentu sistēmas abstrakcija**

Katru no attēlotajiem objektiem un to saiknēm iespējams implementēt pēc vajadzības.

4.2.1. Aģents

Neskatoties uz to, ka aģenti ir ļoti daudzveidīgi, iespējams izvirzīt sekojošas iezīmes – tie ir identificējami objekti, kuriem var piemist rīcība, uztvere un vērtība [4].

Katrs daudzaģentu sistēmas izstrādātājs implementē katru no 4.3 tabulas iezīmēm, kā vien nepieciešams, lai nodrošinātu iecerēto uzvedību.

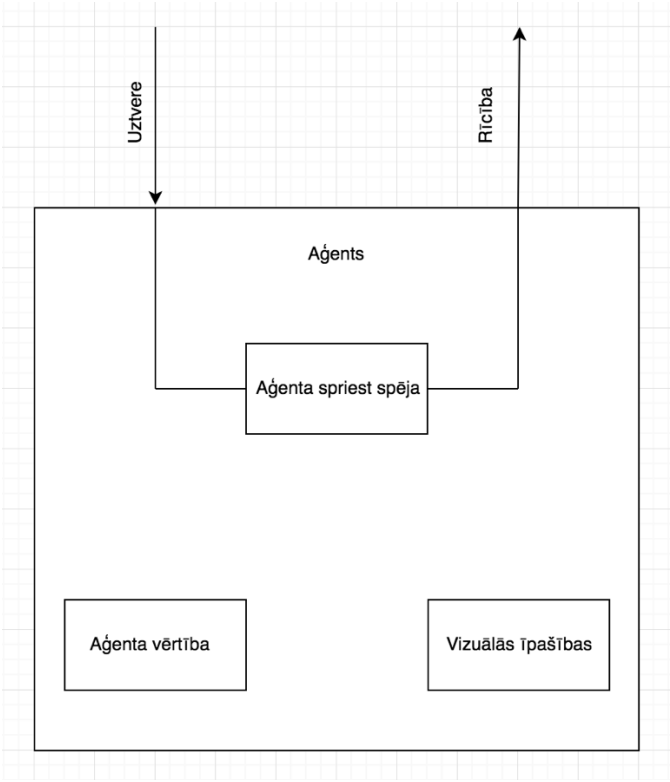
4.3. tabula

Aģenta raksturīgākās iezīmes

Iezīme	**Apraksts**
Uztvere	Spēja apstrādāt vides sniegto informāciju, kā arī potenciāli tikt ietekmētiem no tās. Piemēram, dažādu šķēršļu un citu aģentu atpazīšana vai vides ietekme, kuras rezultātā tiek ietekmēta aģenta pozīcija.
Rīcība	Attēlotā modeļa aģenta darbība pēc informācijas apstrādāšanas. Piemēram, aģenta pozīcijas maiņa vai informācijas apmaiņa ar kādu citu aģentu. Aģents var nebūt fiksēts vienam uzdevumam, tie paši ir spējīgi izvēlēties turpmā darbības soļus, atkarībā no uztvertās informācijas. Aģentu rīcība var izpausties bezdarbībā, piemēram, aģenti var atrasties gaidīšanas režīmā un neko nedarīt, ja

	viņiem tajā brīdī liekas nepieciešami
Vērtība	Šīs ir aģenta īpašības, kuras var netikt vizuāli attēlotas, bet nepieciešamas modeļa būtības attēlošanā. Dzīvo būtņu attēlošanā gadījumā tās varētu izpausties kā noturība pret kādu ietekmējošu faktoru, piemēram, aģenta veselības stāvoklis vai spēja uzņemt ātrumu telpā

Lai vieglāk būtu uztver 4.3. tabulas uzskaitītās iezīmes, tās tiek attēlotas 4.3. attēlā.



4.3. att. **Daudzaģentu sistēmas aģenta uzbūve**

4.2.2. Vide

Daudzaģentu sistēmu vide sastāv no vairākiem fragmentiem, kuri dod priekštatu, ka vide ir viens liels veselums, taču patiesībā tas ir vairāku objektu kopums, kurus arī ir iespējams identificēt un kuriem arī ir īpašības, līdzīgi kā aģentiem.

1:1	1:2	1:3	1:4
2:1	2:2	2:3	2:4
3:1	3:2	3:3	3:4
4:1	4:2	4:3	4:4

4.4. att. **Daudzaģentu sistēmas vides uzbūve**

4.4. Attēlā redzams vides laukums, kurš ir sadalīts vairākos fragmentos. Fragmentu izmēru, to skaitu vai formu, izstrādātājs ir tiesīgs mainīt, lai labāk atspoguļotu modeļa būtibu.

4.3. Vizuālās īpašības

Daudzaģentu sistēmas izstrādātājam ir jānodrošina ne tikai funkcionālo sistēmas pusi, kurā tiek definēta aģenta mijiedarbība ar vidi un tās klātesošajiem, bet arī tās attēlošanu.

Tika izvirzītas attiecīgās īpašības, kuras būtu nepieciešams implementēt un izmantot, lai spētu attēlot kādu no sistēmām. Katrs daudzaģentu sistēmas izstrādātājs implementē katru no šīm iezīmēt, kā vien nepieciešams, lai nodrošinātu iecerēto iznākumu.

4.4. tabula

Aģenta konstatētās vizuālās īpašības un to iespējamās vērtības

Īpašība	**Vērtība**
Krāsa	Neierobežota krāsu palete, aģents var sastāvēt no vairākām krāsām
Tekstūra	Neierobežota tekstūru izvēle, aģents var sastāvēt no vairākām tekstūrām
Forma	Neierobežota formas izvēle
Izmērs	Neierobežots izmērs
Pozīcija	Iespējama pa visu vides laukumu, kā arī ārpus tā
Rotācija	Jebkurā virzienā

4.4. tabulā redzams, ka aģentam krāsa un tekstūra ir vissarežģītākā īpašība, jo tās var būt vairākas, to iespējams nodrošināt ar iepriekš sagatavotiem 2D vai 3D modeļiem, kuriem tiek krāsa un tekstūta uzstādīta nepieciešamajās vietās.

4.5. tabula

Fragmenta konstatētās vizuālās īpašības un to iespējamās vērtības

Īpašība	**Apraksts**
Krāsa	Neierobežota krāsu palete, aģents var sastāvēt no vairākām krāsām
Tekstūra	Neierobežota tekstūru izvēle, aģents var sastāvēt no vairākām tekstūrām
Forma	Kvadrāts
Izmērs	Neierobežots izmērs

4.5. tabulā redzams, ka fragmenta krāsa un tekstūra ir vissarežģītākā īpašība, jo tās var būt vairākas, to iespējams nodrošināt ar iepriekš sagatavotiem 2D vai 3D modeļiem, kuriem tiek krāsa un tekstūta uzstādīta nepieciešamajās vietās.

5. PROTOTIPA REALIZĀCIJA

Šajā sadaļā tiek apskatīta rīka realizācija, tā uzstādījumi, tehnoloģijas izvēle, lietojamība un ievaddatu struktūras iztirzājums ar kura palīdzību iespējams izstrādāt arī kādu citu līdzīgu rīku.

Esošo rīku apskates nodaļā tika secināts, ka konkrētajiem apskatītajiem rīkiem ir būtiskas problēmas, kuras darba autors novērsa ar šo jaunizveidotā rīka prototipa palīdzību. Izstrādātais rīks nav neviena esošā rīka aizvietotājs, bet drīzāk kā papildprogrammatūra - pilnīgi neatkarīga komponente, kuru iespējams varētu izmantot ar citām līdzīgām programmām, kuras piedāvā daudzaģentu sistēmu izstrādi ar priekšnosacījumu, ka tā atbalsta aģentu un vides informācijas ekposrtēšanu katrā simulācijas solī, kādā nolasāmā datnes teksta formātā.

Neviens no apskatītajiem rīkiem nepiedāvā attēlot 3D daudzaģentu sistēmu tīmeklī, tadēļ šis rīks tiek tendēts tieši uz šo platformu.

Katrs no šiem rīkiem piespiež izstrādātāju izmantot konkrētā rīka atbastītās programmēšanas valodas (*Netlogo, Relogo*), kuras diemžēl ir ļoti specifiskas un tendētas tieši uz katra risinājuma platformām, kā arī to mērķis ir atvieglot daudzaģentu sistēmu izstrādes procesu, kas citiem izstrādātājiem var nebūt izdevīgi, jo jāiegulda laiks šo valodu apgūšanā. Jāatdzīst, ka tiek atbalstītas arī plašāk lietotas programmēšanas valodas - *Java* un *Groovy,* taču tas nemaina faktu, ka izstrādātājam ir nepieciešams apgūt kādu no šīm programmēšanas valodām, lai attēlotu daudzaģentu sistēmas. Tādēļ šis rīks tika veidots kā neatkarīga komponente, kura tiešā veidā netiek integrēta ar kādu no apskatītajiem rīkiem. Šī rīka izmantošana neprasa nekādas specifiskas programmēšanas zināšanas vai valodas, taču ar priekšnosacījumu, ka lietotājs spēj izveidot *JSON* teksta datni ar nepieciešamo informāciju par konkrēto simulāciju.

Neviens no apskatītajiem rīkiem nepiedāvā attēlot tikko simulētās darbības, proti, nav iespējams simulāciju patīt atpakaļ, rīka prototipā tas tiek nodrošināts, taču rīka prototipā nav izveidota lietotāju saskarne, kur ērti to varētu lietot.

Abi apskatītie rīki simulāciju un tās attēlošanu veic reālā laikā, kas nozīmē to, ka ierīce uz kuras tiek palaista simulācija pilda nepieciešamās darbības gan simulācijas loģikas veikšanai, gan tās attēlošanai vienlaicīgi, kas pie lielu vai sarežģītu simulāciju gadījumiem varētu izraisīt attēlošanas kvalitātes zudumus - lēnu simulācijas izpildi, tādēļ izstrādātais rīks ir paradzēts lietošanai pēc simulācijas pabeigšanas un datu eksportēšanas.

5.1. Tehnoloģiju izvēle

Lai nodrošinātu rīka uzstādījumu - rīkam nepieciešams attēlot daudzaģentu sistēmas tīmekļa vidē, tika izvēlēts tīmekļa bāzēts ietvars *three.js*, kurš nodrošina 2D un 3D grafisko elementu attēlošanu.

Darba autors neizvirzīja uzstādījumu izvēlēties pēc iespējas ātrdarbīgāku tīmekļa bāzētu ietvaru, taču par galvenajiem kritērijiem izvirzīja ietvara:

- Dokumentāciju
- Nodrošināto funkcionalitāti - 2D, 3D modeļu ielādi, attēlošanu telpā un to manipulēšanu, lai nodrošinātu aģentu un vides vizuālās īpašības, kuras aprakstītas 4.4 un 4.5 tabulās
- *WebGL* atbalstu

Visus augstāk minētos uzstādījumus izvēlētais ietvars spēja nodrošināt, tadēļ netika veikta salīdzināšana ar citiem esošajiem ietvariem, kā arī šis ietvars ir novērtēts par vienu no labākajiem vairākās tīmekļa vietnēs [21, 22].

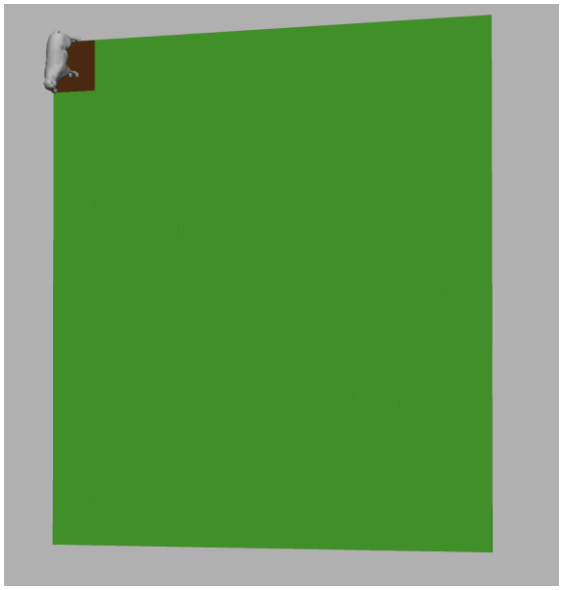
Tīmekļa bāzētam risinājumam ir nepieciešama servera programmatūra, darba autors izmantoja šobrīd vispopulārāko bezmaksas tīmekļa servera programmatūru - *Apache* [23, 24]. Serveris darbojas kā piekļuves vieta tīmekļa vietnei, kura spēj attēlot daudzaģentu sistēmas.

Simulācijas datu glabāšanai tika izmantota *JSON* datne, jo tā ir plaši izmantota un ar lielu tīmekļu pārlūkprogrammu atbalstu, datnes formāts cilvēkam ir viegli salasāms un labojams, kā arī datoriem un citām ierīcēm viegli apstrādājams formāts [25].

5.2. Rīka lietošana

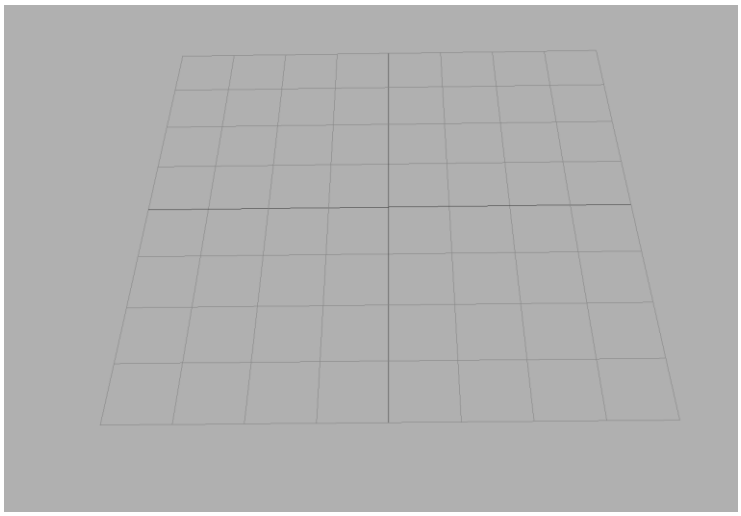
Rīks pats par sevi simulācijas neveic, tikai attēlo tās, tas nozīmē to, ka rīku jāizmanto kopā ar kādu citu programmatūru, kura izpilda simulāciju un spēj nodrošināt rīku ar nepieciešamo informāciju. Pēc ievaddatu izveides nepieciešams tos ielādēt rīkā, ko ar esošo rīka versiju var izdarīt tikai aizvietojot “dati.json” datni, kura atrodas rīka mapē. Dati tiks ielādēti un lietotājam tiks attēlota simulācijas vizualizācija tīmeklī, kur ir iespējams, mainīt skatu, simulācijas ātrumu, kā arī aplūkot simulāciju konkrētā laika momentā.

Nepieciešamības gadījumā datus ir iespējams mainīt ar kādu teksta rediģēšanas programmatūru. Korektru ievaddatu gadījumā tīmekļa vietnē tiks izveidota telpa, par piemēru var aplūkot 5.1 attēlu.



5.1. att. **Izstrādātā rīka vizualizācijas piemērs ar korektiem datiem**

Aģenti un fragmenti tiek izvietoti telpā pēc to koordinātēm. Aģentam ir aitas forma, bet fragments ir 2D plakne, kuru attiecīgi iespējams iekrāsot nepieciešamajā krāsā.



5.2. att. **Izstrādātā rīka vizualizācijas piemērs ar nekorektiem, tukšiem datiem**

Ja simulācijas dati būs tukši vai nekorekti, tad tīmekļa vietnē tiks uzģenerēta tukša telpa, kas redzama 5.2 attēlā.

5.3. Ievaddatu struktūras iztirzājums

Ievaddati tiek balstīti uz 4.4 un 4.5 tabulām, tie sastāv no 2 galvenajām daļām un to apakšvērtībām:

- **settings** - iestatījumu sadaļa, lai lietotājs varētu manipulēt ar vizualizāciju
    - **resolution** - vizualizācijas izšķirtspēja
        - **x** - pixeļu skaits horizontālā virzienā
        - **y** - pixeļu skaits vertikālā virzienā

- **speed** - simulācijas ātrums, katra simulācijas brīza, jeb kadra attēlojums sekundēs
- **fps** - attēloto kadru skaits sekundē, kurš lietotājam tiks parādīts. Šis iestatījums nav saistīts ar simulācijas ātruma iestatījumu

- **animation** - simulācijas datu sadaļa
  - **models** - masīvs ar unikālajām 2D un 3D modeļu datnēm
    - **modeļa unikāla atslēga** - neierobežota teksta formāta atslēga, piemēram, "modelis1"
    - **modeļa datnes atrašanās vieta** - ceļš uz modeļa datni, piemēram, "/models/model3D.obj". Katrs izstrādātājs formātu var mainīt vai pievienot citu.
  - **frames** - masīvs ar simulācijas datiem, katra masīva pievienotā vērtība ir simulācijas datu viens laika sprīdis
    - **fragments** - masīvs ar vides fragmentu informāciju noteiktajā laika brīdī, šeit tiek glabāta visu fragmentu informācija
      - **modelName** - ielādētā modeļa atslēga, kuru nepieciešams attēlot telpā
      - **color** - krāsa fragmentam heksadecimālajā pierakstā, piemēram, "0xFFFFFF". Šo vērtību var norādīt, ja ielādētajam modelim nav krāsa vai nepieciešams to mainīt
      - **position** - fragmenta atrašanās vieta telpā
        - **x** - izvietojums pēc x ass
        - **y** - izvietojums pēc y ass
        - **z** - izvietojums pēc z ass
      - **size** - ielādētā modeļa izmēra koeficients, piemēram, 1, kas nozīmētu to, ka fragmenta izmērs netiek mainīts
      - **texture** - fragmenta tekstūra, šo vērtību var norādīt, ja ielādētajam modelim nav tekstūra vai nepieciešams to mainīt
    - **agents** - masīvs ar aģentu informāciju noteiktajā laika brīdi, šeit tiek glabāta visu aģentu informācija
      - **modelName** - ielādētā modeļa atslēga, kuru nepieciešams attēlot telpā
      - **color** - krāsa fragmentam heksadecimālajā pierakstā, piemēram, "0xFFFFFF". Šo vērtību var norādīt, ja ielādētajam modelim nav krāsa vai nepieciešams to mainīt

- **position** - aģenta atrašanās vieta telpā
    - **x** - izvietojums pēc x ass
    - **y** - izvietojums pēc y ass
    - **z** - izvietojums pēc z ass
- **size** - ielādētā modeļa izmēra koeficients, piemēram, 1, kas nozīmētu to, ka fragmenta izmērs netiek mainīts
- **texture** - aģenta tekstūra, šo vērtību var norādīt, ja ielādētajam modelim nav tekstūra vai nepieciešams to mainīt
- **rotation** - aģenta rotācija telpā
    - **x** - rotācija pēc x ass
    - **y** - rotācija pēc y ass
    - **z** - rotācija pēc z ass

5.4. Ievaddatu apstrāde

Pēc tīmekļa vietnes palaišanas, tā ielāde simulācijas datus, datu ielādes pirmkoda paraugu iespējams apskatīt 4. pielikumā. Lai līdzīgiem aģentiem, kur forma neatšķiras, bet citi atribūti, piemēram, krāsa, nebūtu jāizveido jauns 2D vai 3D modelis, ielādētajai formai tiek uzstādīta krāsa, izmērs, rotācija, tādā veidā var iegūt vienu formu ar dažādiem atribūtiem. Prototipa izveidē netika implementēta tekstūras piešķiršana. Visas šīs īpašības izņemot pozīciju, teorētiski būtu iespējams ielādēt arī tikai no 2D un 3D modeļa, ja nepieciešams attēlot sarežģītāku objektu. Pēc datu ielādes vietne ielādē aģentu un vides fragmentu modeļu ģeometriju, kura pirmkoda paraugs apskatāms 5. pielikumā. Kad visi dati ielādēti, tiek izveidota 3D telpa, kurā iespējams manipulēt ar skatu. Lai būtu iespējams simulāciju aplūkot jebkurā laika momentā, nepieciešams izveidot funkciju, kura attēlo datus atkarībā no izvēlētā laika momenta.

```
function prepareFrame(frameIndex) {
  //Prepare fragments
  prepareFragments(frameIndex)

  //Prepare agents
  prepareAgents(frameIndex)
}
```

5.3. attēls. Simulācijas datu sagatavoša konkrētā laika momentā

5.3. att. redzama funkcija, kura izveido nepieciešamo fragmentu un aģentu izvietojumu telpā izvēlētajā laika momentā, ņemot vērā arī tā vizuālās īpašības. 6. pielikumā apskatāms pirmkoda paraugs, kur aģentiem šīs īpašības tiek uzstādītas.

REZULTĀTI

Rezultātā izpildīti visi izvirzītie darba mērķi. Tika iegūta pamatstruktūra ar kuras palīdzību iespējams attēlot plašu klāstu daudzaģentu sistēmu. Tā kalpo kā piemērs pēc kura vadoties iespējams izstrādāt līdzīgu rīku kādā citā vidē, kas varētu nebūt tīmeklis.

Kā arī rezultātā tika izstrādāts tīmeklī bāzēts daudzaģentu sistēmu attēlošanas rīka prototips. Rīka uzstādījumi ir sasniegti - iespējams attēlot plašu klāstu sistēmu, kā arī rīka pilnā versija novērstu esošo, aplūkoto daudzaģentu sistēmu attēlošanas rīku nepilnības, prototipā nav izstrādāta funkcionējoša lietotāja saskarne. Rīkam ir sekojošas funkcijas: iepriekš sagatavotu daudzaģentu simulācijas datu ielāde; 2D,3D modeļu ielāde un to vizuālo īpašību manipulācija; ielādēto modeļu izvietojums 3D telpā pēc ielādēto datu vērtībām; simulācijas datu attēlojums animācijas veidā; simulācijas konfigurēšana pirms tās palaišanas - izšķirtspējas maiņa, attēloto kadru skaita maiņa; simulācijas lietošana pēc datu ielādes - ātruma maiņa, apturēšana, manipulēšana ar skatu (rotācija ap orbītu, pietuvināšana un attālināšana) kamēr simulācija tiek attēlota gan kad simulācija ir apturēta.

Neskatoties uz to, ka izstrādātais rīks novērš aplūkoto rīku nepilnības, tas nenodrošina vidi, kurā lietotājam būtu iespējams pilnvērtīgi pētīt daudzaģentu sistēmas, tā mērķis ir attēlot aģentus un vidi. Tas nozīmē to, ka simulācijas dati, kurus nav paradzēts vizuāli attēlot telpā netiek izmantoti, līdz ar to, tos aplūkot nav iespējams. Šo vērtību attēlošanā būtu nepieciešams izveidot citu rīku vai papildināt esošo, kas būtu iespējams.

SECINĀJUMI

Iepazīstoties ar daudzaģentu sistēmām, darba autors secināja, ka aģentu bāzēta modelēšana ir ļoti noderīga dažādu grupveida sabiedrību pētīšānā, jo problēmu iespējams sadalīt mazākās problēmās - aģentos. Ar tiem manipulējot simulācijas kontekstā, iespējams novērot rezultātus, kurus savadāk varbūt nebūtu iespējams paredzēt.

Analizējot dažādas daudzaģentu sistēmu realizācijas, darba autors secināja, ka vizualizācijas galvenais uzstādījums ir sniegt lietotājam pēc iespējas vienkāršāku attēlojumu, lai vieglāk būtu uztvert modeļa galveno domu - izdaiļošana tikai traucē. Vel tika secināts, ka to pielietojums neaprobežojas tikai ar problēmu pētišanu, tās tiek izmantotas arī dažādu jau atrisinātu problēmu un scenāriju attēlošanā. Tākā aģents ir ļoti abstrakts, izstrādātājam ir viegli panākt nepieciešamo uzvedību, tādējādi tā pielietojums ir ļoti daudzveidīgs.

Aplūkojot esošos daudzaģentu sistēmu izstrādes rīkus, darba autors secināja, ka tie nodrošina nepieciešamo funkcionalitāti un tehnoloģijas, lai izstrādātājs krietni īsākā laika posmā spētu modelēt savu scenāriju, taču pašas simulācijas pētīšana ar šo rīku palīdzību varēja būt labāka. Abu rīku izstrādātāji vēsta, ka šie rīki tiek izstrādāti jau vairāk nekā 10 gadus, kas nozīmē to, ka šāda veida rīku izveide visticamāk nav triviāla.

Aplūkojot izvirzītās tīmekļa vizualizācijas tehnoloģijas, darba autors secināja, ka tīmekļu tehnoloģijas turpina attīstīties, ja agrāk interaktīvu vietņu aplūkošanai bija vajadzīga papildus programmatūra uz ierīces, tad tagad tas vairs nav nepieciešams. Tika arī secināts, ka ļoti uzmanīgi jāseko līdz izstrādātās programmatūras uzturēšanai, jo pat ja pēc izstrādes veikšanas izveidotā programmatūra ir droša, tas nenozīmē to, ka tā tāda paliks. Izmantotās tehnoloģijas var mainīties vai novecot, radot drošības problēmas.

Pēc rīka izstrades, darba autors secināja, ka ar viena rīka palīdzību attēlot dažādas daudzaģentu sistēmas ir ļoti sarežģīti un iespējams pat neiespējami, jo nav paradzema modeļa jēga un nepieciešamais rezultāts.

Pētījumu iespējams turpināt, aplūkojot vairāk daudzaģentu sistēmu realizācijas. Neskatoties uz to, ka izstrādātais rīks spēj attēlot plašu skaitu sistēmu un to, ka vizualizācijai jābūt pēc iespējas vienkāršakai, lai pēc iespējas vieglāk būtu saprast modeļa jēgu, darba autors secināja, ka teorētiski varētu pastāvēt sistēmas, kur aģentiem nepieciešams attēlot dinamisku, neparadzemu un vienmērīgu formas maiņu animācijas veidā, kā arī dažādu informatīvu vērtību attēlošana vizualizācijas telpā, tas ir, dinamiski skaitļi un teikumi, kurus būtu ļoti nepraktiski attēlot ar 2D un 3D modeļu palīdzību. No tā var secināt kādēļ esošie rīki piedāvā aģentu vizuālo izskatu mainīt ar programmēšanas valodu palīdzību. Šos izņēmumu gadījumus nepieciešams izpētīt pirms varam secināt, ka šāda universāla rīka izstrāde ir iespējama.

Bakalaura darba izstrādes beigās, darba autors secināja, ka izstrādātais bakalaura darbs citiem var palīdzēt iepazīties ar daudzaģentu sistēmām un to pielietojumiem, kā arī ar to izstrādes iespējām, izmantojot aplūkotos rīkus vai mēģinot realizēt un attēlot savu scenāriju tīmeklī. Kā arī iedvesmot lasītājus izmantot aģentu bāzētu modelēšanu, lai atrisinātu kādu viņam aktuālu problēmu.

Autors darba gaitā ir ieguvis vērtīgas zināšanas dažādu problēmu risināšanā izmantojot aģentu bāzētu modelēšanu, sapratis simulāciju veikšanas nepieciešamību, ieguvis pamatzināšanas to veidošanā un attēlošanā.

IZMANTOTIE AVOTI UN LITERATŪRA

1. Design Guidelines for Agent Based Model Visualization [tiešsaiste] – [atsauce: 10.05.2018.]. Pieejams: http://ccl.northwestern.edu/papers/2009/Kornhauser,Wilensky&Rand_DesignGuidelinesABMViz.pdf
2. Agent-based Modeling [tiešsaiste] – [atsauce: 13.05.2018.]. Pieejams: https://www.comses.net/about/faq/
3. Introduction to Agent-Based Modeling [tiešsaiste] – [atsauce: 13.05.2018.]. Pieejams: https://soc.kuleuven.be/ceso/historischedemografie/resources/pdf/Grow.pdf
4. Agent-based Modeling And Simulation [tiešsaiste] – [atsauce: 16.05.2018.]. Pieejams: https://www.informs-sim.org/wsc09papers/009.pdf
5. Daudzaģentu sistēmu pielietojumi datortīklu starpprogrammatūrā [tiešsaiste] – [atsauce: 16.05.2018.]. Pieejams: https://ortus.rtu.lv/science/lv/publications/4581
6. NetLogo User Manual [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://ccl.northwestern.edu/netlogo/faq.html
7. NetLogo Web [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://www.netlogoweb.org/info
8. Repast: A Software Toolkit for Agent-Based Social Science Modeling [tiešsaiste] – [atsauce: 23.05.2018.]. Pieejams: http://www2.econ.iastate.edu/tesfatsi/repastsg.htm
9. Repast Simphony [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://repast.github.io/repast_simphony.html
10. Repast Simphony Frequently Asked Questions [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://repast.sourceforge.net/docs/RepastSimphonyFAQ.pdf
11. Repast Simphony: Freezedrying and Loading Data [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://sourceforge.net/p/repast/mailman/attachment/D032535911F942B8840FABC16CAD79C0@ENCLWATTS/2/
12. Here's How Many Internet Users There Are [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://time.com/money/3896219/internet-users-worldwide/
13. Mobile phone internet user penetration worldwide from 2014 to 2019 [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams:

https://www.statista.com/statistics/284202/mobile-phone-internet-user-penetration-worldwide/

14. GPU Rendering vs. CPU Rendering [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://blog.boxx.com/2014/10/02/gpu-rendering-vs-cpu-rendering-a-method-to-compare-render-times-with-empirical-benchmarks/
15. 3D Graphics on the Web: a Survey [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://pdfs.semanticscholar.org/b048/98eaf36c3aee9d10f51e9bace9a09ce9cf14.pdf
16. HTML5 Canvas [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://www.w3schools.com/html/html5_canvas.asp
17. Why you should learn WebGL [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://www.pluralsight.com/blog/software-development/webgl-basics
18. A collection of WebGL frameworks and libraries [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://gist.github.com/dmnsgn/76878ba6903cf15789b712464875cfdc
19. Deploying an Applet [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://docs.oracle.com/javase/tutorial/deployment/applet/deployingApplet.html
20. Lesson: Deployment In-Depth [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://docs.oracle.com/javase/tutorial/deployment/deploymentInDepth/index.html
21. 8 Best 3D Javascript Libraries [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: http://bashooka.com/coding/8-best-3d-javascript-libraries/
22. The Top 3D JavaScript Libraries For Web Developers [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://1stwebdesigner.com/3d-javascript-libraries/
23. Web Server Usage Statistics [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://trends.builtwith.com/web-server
24. Usage of web servers for websites [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://w3techs.com/technologies/overview/web_server/all
25. JavaScript Object Notation [tiešsaiste] – [atsauce: 18.05.2018.]. Pieejams: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

Pielikumi

1. pielikums. **WebGl integrēšanas piemērs**

HTML Canvas elementa pievienošana:

```
<body>
  <canvas id="glCanvas" width="640" height="480"></canvas>
</body>
```

WebGL integrācija un izmantošana:

```
const canvas = document.querySelector("#glCanvas");
// Initialize the GL context
const gl = canvas.getContext("webgl");

// Only continue if WebGL is available and working
if (!gl) {
  alert("Unable to initialize WebGL. Your browser or machine may not support it.");
  return;
}

// Set clear color to black, fully opaque
gl.clearColor(0.0, 0.0, 0.0, 1.0);
// Clear the color buffer with specified clear color
gl.clear(gl.COLOR_BUFFER_BIT);
```

JNLP datnes paraugs:

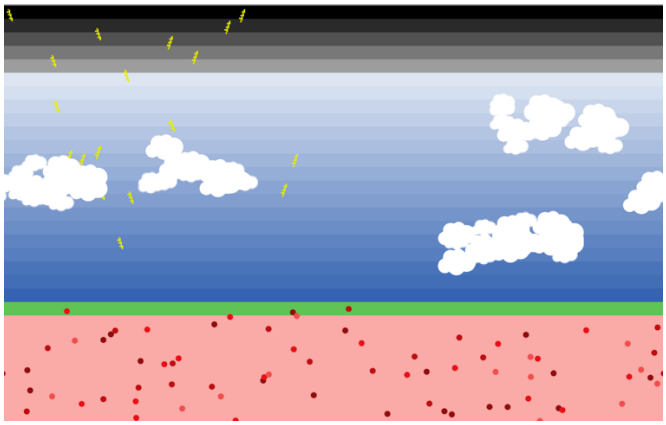
```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="" href="">
    <information>
        <title>Dynamic Tree Demo</title>
        <vendor>Dynamic Team</vendor>
    </information>
    <resources>
        <!-- Application Resources -->
        <j2se version="1.7+"
            href="http://java.sun.com/products/autodl/j2se" />
        <jar href="DynamicTreeDemo.jar" main="true" />

    </resources>
    <applet-desc
         name="Dynamic Tree Demo Applet"
         main-class="components.DynamicTreeApplet"
         width="300"
         height="300">
     </applet-desc>
     <update check="background"/>
</jnlp>
```

Norāde uz *JNLP* datni:

```
<body>
  <!-- ... -->
  <script src="https://www.java.com/js/deployJava.js"></script>
  <script>
    var attributes = {
      code: 'components.DynamicTreeApplet',
      width: 300,
      height: 300
    };
    var parameters = {
      jnlp_href: 'dynamictree_applet.jnlp'
    };
    deployJava.runApplet(attributes, parameters, '1.7');
  </script>
  <!-- ... -->
</body>
```

3. pielikums. **Daudzaģentu sistēmu vizuālo īpašību novērtējums**

Klimats



Apraksts par modeli

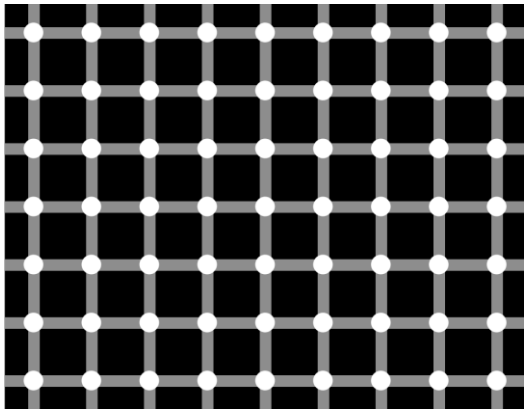
Sistēma, kurā ar aģentu palīdzību tiek attēlota klimata maiņa

Vides iezīme	**Apraksts**
Fragmentēta	Laukums ir sadalīts mazākās, bet vienādās vienībās, to var secināt pēc fona krāsu toņa maiņas
Vienības forma	Vizuāli pateikt nevar, kvadrāts vai taisnstūris, taču ieskatoties realizācija un mainot kāda fragmenta krāsu, var ievērot, ka tas ir kvadrāts
Krāsa	Fragmentam ir viena krāsa. Melna, pelēka, zaļa, sarkana utml.
Fragmenta izmērs	Konstants
Fragmenta pozīcija	Nemainīga, fons nepārvietojas
Rotācija	Nav
Fragmentu identificējamība	Neanalizējot realizāciju, to nav viegli izdarīt

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Mākonis, zibens, aplis
Krāsa	Viena veida krāsa, balta, sarkana, dzeltena

Izmērs	Konstants
Pozīcija	Mainīga, kustību neveic tikai pa fragmenta centriem
Rotācija	Iespējama visos virzienos
Identificējamība	Iespējama, ja aģents redzams kadrā

Mākslas ilūzija



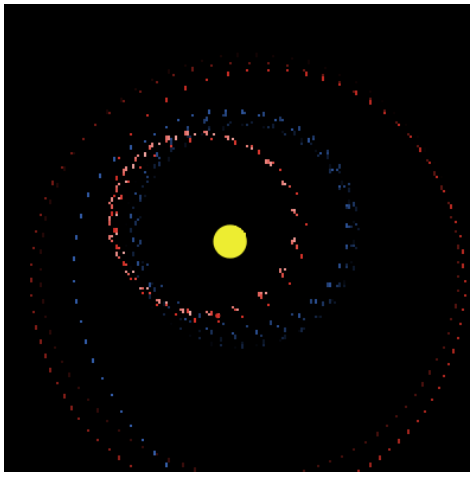
Apraksts par modeli

Sistēma, kurā tiek attēlota optiska ilūzija

Vides iezīme	**Apraksts**
Fragmentēta	Laukums ir sadalīts mazākās vienībās, kuras, krāsu dēļ, dod iespaidu, ka vide sastāv no lielākām vienībām vai citādākām formām nekā patiesībā
Vienības forma	Kvadrāts
Krāsa	Pelēka, melna, atkarībā no pozīcijas
Fragmenta izmērs	Konstants, taču nav viegli izmērams, jo ir grūti identificējams
Fragmenta pozīcija	Nemainīga
Rotācija	Nav
Fragmentu identificējamība	Sarežģīta, jo tie saplūst kopā

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Aplis
Krāsa	Balta
Izmērs	Konstants, izmērāms
Pozīcija	Nemainīga, katram aģentam ir tikai viena pozīcija
Rotācija	Nav
Identificējamība	Iespējama, aģents nepārklāj citu aģentu un ir labi redzami

Gravitācija



Apraksts par modeli

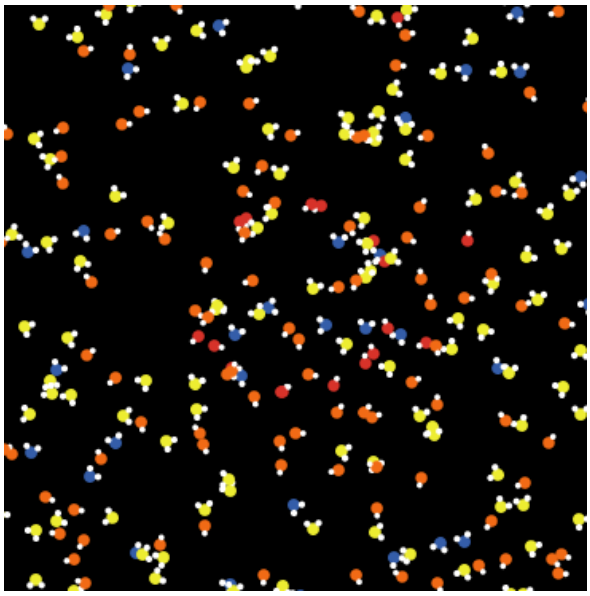
Sistēma, kurā tiek attēlota gravitācijas spēks starp objektiem

Vides iezīme	**Apraksts**
Fragmentēta	Laukums ir sadalīts mazākās, bet vienādās vienībās, to var izsecināt pēc aģenta atstātajām pēdām
Vienības forma	Kvadrāts, jo krāsas maiņas brīdī tam ir šada forma
Krāsa	Vairāki toņi no zilas, sarkanas, kā arī melna, kas dod iespaidu, ka fragments izzūd. Krāsa lēnām atgriežas uz melnu krāsu, ja aģents to neiespaido.

Fragmenta izmērs	Konstants
Fragmenta pozīcija	Nemainīga, pēc fragmentu krāsas maiņas to var novērot
Rotācija	Nav
Fragmentu identificējamība	Sākumā sarežģīta, bet kad aģents tos ietekmē, tad tā pārtop par viegli novērojamu īpašību

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Aplis
Krāsa	Katram aģentam ir sava krāsa, zila, sarkana, dzeltana un tā ir nemainīga.
Izmērs	Konstants, visi aģenti nav vienādā izmērā
Pozīcija	Mainīga, katram aģentam ir tikai viena pozīcija. Var izsecināt to, ka aģents var pārvietoties izlaižot fragmentus
Rotācija	Pēc formas nav nosakāma
Identificējamība	Iespējama, aģents nepārklāj citu aģentu un ir labi redzami

Ķīmiska reakcija



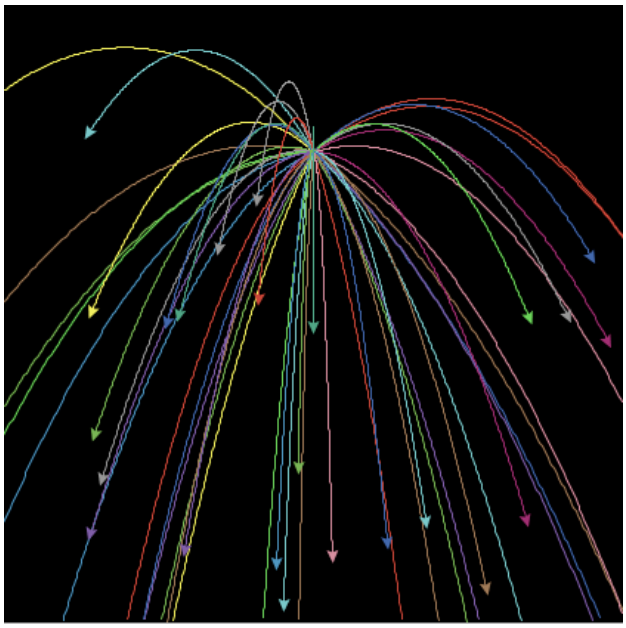
Apraksts par modeli

Sistēma, kurā tiek attēlota ķīmiska reakcija starp molekulām

Vides iezīme	**Iezīmes apraksts, vērtība**
Krāsa	Melna
Fragmentu Identificējamība	Nav iespējama, vide varētu būt viens liels fons

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Dažādu molekulu reprezentācija, mainīga
Krāsa	Aģents var sastāvēt no vairākām krāsām
Izmērs	Konstants
Pozīcija	Mainīga, vienmērīga, aģentam var būt tikai viena pozīcija
Rotācija	Mainīga, visos virzienos
Identificējamība	Iespējama, taču aģents var aizsekt otru aģentu

Daļiņu specefekts



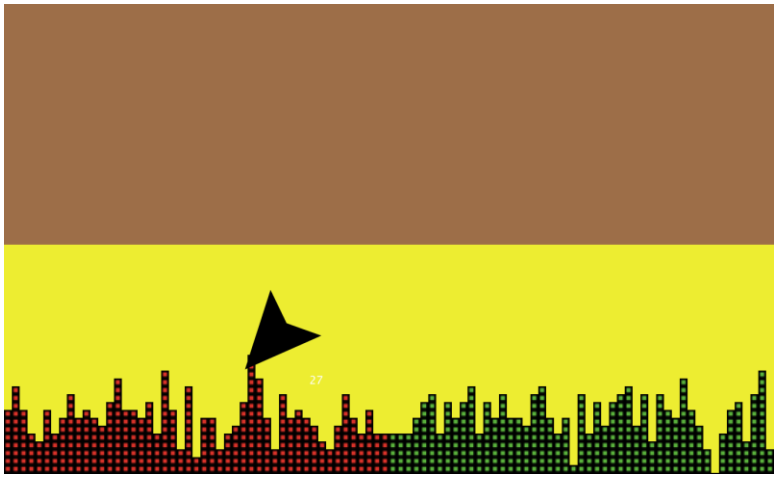
Apraksts par modeli

Sistēma, kurā tiek veidots specefekts izmantojot aģentus

Vides iezīme	**Apraksts**
Fragmentēta	Laukums ir sadalīts mazākās, bet vienādās vienībās, to var izsecināt pēc aģenta atstātajām pēdām
Vienības forma	Kvadrāts, jo krāsas maiņas brīdī tam ir šada forma
Krāsa	Dažādas krāsas
Fragmenta izmērs	Konstants
Fragmenta pozīcija	Nemainīga, pēc fragmentu krāsas maiņas to var novērot
Rotācija	Nav
Fragmentu identificējamība	Sākumā sarežģīta, bet kad aģents tos ietekmē, tad tā pārtop par viegli novērojamu īpašību

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Trijstūris
Krāsa	Katram aģentam ir sava krāsa, tā ir nemainīga.
Izmērs	Konstants, visi aģenti vienādā izmērā
Pozīcija	Mainīga, no fragmenta uz fragmentu, katram aģentam ir tikai viena pozīcija
Rotācija	Iespējama visos virzienos
Identificējamība	Iespējama, ja aģents neaizsedz citu aģentu

Nejaušība



Apraksts par modeli

Sistēma, kurā aģents ar nejaušības principu ietriecās apakšējā malā, nomainot fragmenta krāsu.

Vides iezīme	**Apraksts**
Fragmentēta	Laukums ir sadalīts mazākās, bet vienādās vienībās, to var secināt pēc aģenta ietriekšanās kādā no malām
Vienības forma	Kvadrāts, jo krāsas maiņas brīdī tam ir šada forma
Krāsa	Dažādas krāsas
Fragmenta izmērs	Konstants
Fragmenta pozīcija	Nemainīga, pēc fragmentu krāsas maiņas to var novērot
Rotācija	Nav
Fragmentu identificējamība	Sākumā neiespējama, bet kad aģents tos ietekmē, tad tā pārtop par viegli novērojamu īpašību

Aģenta iezīme	**Iezīmes apraksts, vērtība**
Forma	Bulta
Krāsa	Melna, nemainīga

Izmērs	Konstants
Pozīcija	Mainīga, kustību veic no fragmenta uz fragmentu, aģentam ir tikai viena pozīcija
Rotācija	Iespējama visos virzienos
Identificējamība	Iespējama, ja aģents redzams kadrā

```
function loadSimulationData() {
  fileLoader.load(
    // resource
    'hk14004/data.json',
    // onLoad callback
    function(data) {
      // output the text to the console
      console.log("Started parsing simulation!")
      jsonData = JSON.parse(data)
      console.log("Finished parsing simulation!")
      // Extract Settings
      settings = jsonData["settings"]
      frameX = settings["resolution"]["x"]
      frameY = settings["resolution"]["y"]

      // Extract Animation
      animation = jsonData["animation"]
      models = animation["models"]
      frames = animation["frames"]

      // Load models all at once
      loadAllModels(models, function() {
        debugLog("All unique models loaded")
        initWorld();
        startAnimation();
      })
    },
    // onProgress callback
    function(xhr) {
      console.log((xhr.loaded / xhr.total * 100) + '% loaded');
    },
    // onError callback
    function(err) {
      console.error('An error happened');
    }
  );
}
```

```
async function loadAllModels(modelsSourceDict, completionHandler) {
  var loader = new THREE.BufferGeometryLoader();
  for (var modelName in modelsSourceDict) {
    var pathToModel = modelsSourceDict[modelName];
    var modelIsLoaded = new Promise(
      function(resolve, reject) {
        loader.load(
          // resource URL
          pathToModel,
          // onLoad callback
          function(geometry) {
            loadedModelsDict[modelName] = geometry
            debugLog("Loaded " + modelName + " model")
            resolve(geometry)
          },
          // onProgress callback
          function(xhr) {
            console.log((xhr.loaded / xhr.total * 100) + '% loaded');
          },
          // onError callback
          function(err) {
            console.log('An error happened');
          }
        );
      }
    );

    await modelIsLoaded
      .then(function(fulfilled) {})
      .catch(function(error) {});

  }
  completionHandler()
}
```

```
function prepareAgents(frameIndex) {
  let agentsInfo = frames[frameIndex]["agents"]

  // Loop agents and prepare them
  for (var i = 0; i < agentsInfo.length; i++) {
    // Create agent based on geometry
    let modelName = agentsInfo[i]["modelName"]
    let loadedModelGeo = loadedModelsDict[modelName]

    var material = new THREE.MeshLambertMaterial({
      color: parseInt(agentsInfo[i]["color"])
    });
    var agentModel = new THREE.Mesh(loadedModelGeo, material);

    // Position
    agentModel.position.x = agentsInfo[i]["position"]["x"]
    agentModel.position.y = agentsInfo[i]["position"]["y"]
    agentModel.position.z = agentsInfo[i]["position"]["z"]

    //Orientation
    agentModel.rotateX(agentsInfo[i]["orientation"]["x"])
    agentModel.rotateY(agentsInfo[i]["orientation"]["y"])
    agentModel.rotateZ(agentsInfo[i]["orientation"]["z"])

    //Size
    let size = agentsInfo[i]["color"]
    agentModel.geometry.scale(size,size, size);

    //Adding to the scene
    scene.add(agentModel)
    renderedObjects.push(agentModel)
  }
}
```

Bakalaura darbs „Daudzaģentu sistēmas vizualizācija” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Hardijs Ķirsis

Rekomendēju/nerekomendēju darbu aizstāvēšanai
Darba vadītājs: profesors Dr. dat. Guntis Arnicāns _____ 28.05.2018.

Recenzents: Dr.dat., prof. Jānis Bičevskis

Darbs iesniegts Datorikas fakultātē 28.05.2018.
Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe_____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē
__.__.2018. prot. Nr. ____.
Komisijas sekretārs: