Проект „Симулация на град“

Стойко Кръстев : 2401697014

Въведение

Целта на проекта е да се създаде симулация на град, изпълнен с граждани, които представляват агенти, чието поведение зависи от индивидуалните характеристики на индивидите, взаимоотношенията им с други граждани и от тяхната текуща локация и занаят. Симулацията ще бъде базирана на ходове, като всеки ход идейно символизира около 15 мин време. В началото на всеки ход координиращият агент CityAdmin уведомява всички активни агенти граждани за началото на нов ход. Когато агентите граждани започнат хода си, те имат право на няколко малки действия и едно голямо действие, след което сигнализират на координатора за приключването на хода си. След като всички агенти са приключили действията си, администратора „придвижва“ света напред, извършва планирани действия за край на ход ако има такива и започва нов ход, ако потребителят, управляващ симулацията, е натиснал бутон за нов ход или е задал поредица от ходове. Това което желаем да видим в симулацията е органичното и естествено развитие на отделните индивиди в града. Как хората живеещи наблизо, работещи на еднакви или близки места, ходещи по един и същ път или посещаващи еднакви паркове се запознават и след време стават приятели. Как човек е по-вероятно да работи някъде ако е бил препоръчан от приятел работещ на същото място. Как това се изменя от личните качества и характер на всеки индивид. В сегашната си форма проектът представлява основа върху която може да се изгради много по-сложна и мащабна симулация, но за този момент е работеща базова симулация на града и хората с някои основни симулационни функционалности и работеща логика за комуникация между агенти, онтология и база данни.

Нека прегледаме по-подробно имплементациите на отделните компоненти на симулацията и идеите зад тях. Те ще бъдат логически разделени на : Структура на онтология, База данни, Агенти, Комуникация

между агенти, Потребителски интерфейс, Онтология по време на симулация, Цялостната система.

Структура на онтология

Онтологията е създадена чрез Protege и съдържа класовете Job, Person, Place и Possession като Place има подкласовете: Apartment, Factory, Home, OtherPlace, Shop, Square. От тях в употреба реално са само Job, Person и Place, докато останалите са дефинирани за бъдещо развитие, но в този момент не се използват. Подобни дефинирани, но незавършени елементи има и при object и data properties. Трите класа, който са в употреба, са необходими, за да отговорим на въпроси като „Кой познава кого?“, „Кои хора са приятели?“, „Кой къде и какво работи“ „На кои места се предлага тази работа“. Неизползваните класове ще се използват за допълнително конкретизиране и логически съждения в бъдещо развитие на проекта.

Object properties описват отношенията между индивиди от класовете. Дефинирал съм следните отношения като само hasPossession не е в употреба.

worksAt – функционална връзка между човек и място която показва къде работи човекът. Човек може да работи само на едно място в даден момент.

isStaffedBy – функционална връзка между място и работа, показваща че на мястото се практикува описаната работа. На едно място се практикува само един занаят.
hasOccupation – връзка между човек и работа която показва че човек в момента работи на определена работа/позиция. Добавено е и логическо съждение: “worksAt o isStaffedBy SubPropertyOf: hasOccupation” което гласи че ако човек работи на дадено място и на това място се практикува даден занаят то следва че човекът работи тази работа.

houses – връзка между място и човек която показва че мястото подслонява този човек. Инверсно свойство на livesIn.

livesIn – връзка между човек и място която показва че човек живее на дадено място. Инверсно свойство на houses.

isFriendsWith – симетрична връзка между човек и човек. Обозначава приятелство.

isLocatedAt – функционална връзка между човек и място. Всеки човек винаги е някъде и може да бъде само на едно място в даден момент.

isOwnedBy – връзка между място и човек показваща че мястото е притежавано от човека. Инверсно свойство на owns.

owns – връзка между човек и място показваща че човек притежава мястото. Инверсно свойство на isOwnedBy.

knows – симетрична връзка между човек и човек обозначаваща че хората се познават.

Поради идеите за бъдещо развитие и сложност на симулацията на проекта, съм дефинирал голям брой Data properties и няма да изброявам всичките. Тези, които се отнасят за Person са най-многобройни и включват следните: hasName, hasPsychologicalTrait, hasSkilllevel, hasPhysicalTrait. Като свойствата без hasName имат общо 27 подсвойства описващи умения, характер и външен вид. Повечето от тях не оказват влияние на симулацията, но все пак към онтологията се създават и добавят индивиди които ги притежават, просто не се използват по време на симулация все още.

Свойствата описващи Place са по-малко и те са: hasName, hasLocalSalaryModifier, hasProductionPerTurn, hasShifts, isLocatedAtX, isLocatedAtY, hasStaffSlots. От тях всички се създават динамично в началото на симулацията като hasProductionPerTurn, hasLocalSalaryModifier и hasName не оказват никакво влияние на действието все още.

Свойствата описващи Job са hasName, hasSalary и requiresSkillLevel, като от тях requiresSkillLevel има подсвойства съответстващи на подсвойствата на hasSkillLevel, които описват всеки човек. Идеята е че за да работят дадена работа трябва минималните умения да бъдат покрити.

Това е текущата базова структура на онтологията.

База данни

Базата данни е h2 и се съхранява локално. Служи за персистентност на данните. Поради естеството на симулацията и фактът, че всяка симулация бързо и непредвидимо се изпълва с информация, смятам че е най-адекватно да се запазват само данни за началните условия на симулацията като профили на граждани и места. Текущо е имплементирано съхранението на информация за граждани като се пази информацията която трябва да се зареди в онтологията, а това е информацията съответстваща на data свойствата hasName, hasPsychologicalTrait, hasSkilllevel, hasPhysicalTrait. Реално hasPsychologicalTrait, hasSkilllevel и hasPhysicalTrait са имената на групи от свойства и тяхната информация се запазва в отделни таблици, докато главната таблица за Човек има име и ключове сочещи към другите таблици.

Агенти

Координатор/администратор: Отговаря за контролиране на симулацията. Този агент стартира симулацията като създава града, хората , местата, занаятите и инициализира агентите. Той служи и за координатор на ходовете на симулацията, защото подава сигнал за началото на всеки нов ход, а агентите на гражданите извършват определен брой действия само ако са получили сигнал за нов ход. Освен тези длъжности за контролиране на началото и хода на симулацията този агент осъществява и връзката с онтологията. Искам да има достъп до онтологията само от едно място, което има своите минуси и плюсове, но за началната симулация предпочитам централизиран контрол. Логиката при създаването на отделните обекти в симулацията било то хора, места или занаяти е сходна и доста неконтролирана. Използва се Random класът в съвкупност с помощни enum класове или switch case където е нужно, за да се инициализират хора, места и занаяти с характеристики в дадени диапазони и стойности сред определени възможни.

Граждани: Те са движещите се елементи в симулацията. Всеки Person е гражданин, който на всеки ход извършва по едно голямо и няколко малки действия. След като извърши тези движения, той сигнализира на координатора че е готов. Когато получи сигнал за нов ход той отново прави

няколко действия и приключва хода си. Винаги в началото на всеки нов ход гражданинът си опреснява знанията за околността като пита координатора какво има покрай него. Това запитване е нужно защото само координатора работи с онтологията, а само от там може да се изведе актуална информация за текущото обкръжение на даден гражданин. На този етап възможното голямо действие е са само едно и това е движение към съседно място което е улица. Изборът за този момент е на случаен принцип. По интересни са малките действия от които има няколко: attemptFriendsship, askForMap, relax и thinkOfWork.

attemptFriendship представлява опит на агента да създаде приятелство и има 4 възможни резултата. Ако в момента когато е взел решението за опит за сприятеляване няма никои около действащия агент, то той ще получи стрес и процесът ще приключи. Стресът е статистика притежавана от всеки гражданин, която за момента само се повишава и понижава от различни действия, но целта е при бъдещо развитие тя да влияе на избора на действия. Ако около агента има хора, то той избира един от тях на произволен принцип и се опитва да се сприятели с тях. Ако те вече са приятели стресът на агента намаля, но ако не са били се взема решение от другия агент дали да приеме предложението за приятелство. Решението е повлияно от досегашната бройка приятели на агента, статистиките му, както и няколко характеристики и черти. При успех двата агента стават приятели. При неуспех няма негативен резултат.

askForMap е действие при което агент иска друг агент да сподели своята информация за известната му карта или по-точно местата, които е видял. Трябва да отбележа, че в началото на симулацията гражданите не знаят цялата карта, а в началото на всеки ход „виждат“ кои места и кои хора са около тях като така трупат информация в списък с известни места. Проверките са сходни на attemptFriendship като възможните изходи са : ако агента е сам, успех и неуспех.

Relax е действие което олицетворява как гражданинът се наслаждава на града. В повечето случаи това действие ще намали стресът на агента, но в 30% от случаите ще го повиши.

Последното имплементирано малко действие е thinkOfWork. Агентът внезапно се притеснява за работата си. Ако вече има работа се „успокоява“ и намаля стреса си, ако няма работа, то той подава молба да започне работа на всичките му известни места. Действието се обработва от администратора на града като се преглежда списъкът от известни на агента места и се премахват тези без свободни позиция и тези чиито минимални изисквания за позиция не са покрити от уменията на агента. В този момент ако резултатният списък не е празен се избира първият елемент и гражданинът се записва че работи там, В бъдещото развитие на проекта мястото ще се избира въз основа на личните предпочитания на гражданина за тип работа и заплата, както и дали има приятели работещи там.

Комуникация

В града ще има доста агенти и те трябва да комуникирам помежду си, това ще се извършва чрез Agent Communication Language или ACL съобщения. Всеки агент има CyclicBehaviour където слуша за идваща комуникация от други агенти като разпределя съобщенията и отговаря главно въз основа на performative(Agree,Inform..) и conversationId като понякога е от значения и изпращача на съобщението. Гражданите получават съобщения от координатора/аднимистратора за start-of-turn, receive-salary, new-job като те са прости INFORM съобщения, които не се нуждаят от отговор. При получаване на съобщение с conversationId request-known-places или request-friendship ситуацията е малко по-интересна. При заявката за места агента първо решава дали да отговори, след което ако реши да отговори изпраща потвърждение, че ще отговори на запиталия го агент и след това изпраща съобщение съдържащо масив с известните му места. Ако не желае да сподели изпраща REFUSE съобщение. Съобщенията от тази комуникация от другия край се четат в поведението RequestKnownPlacesFromB. По време на писане на документа разбрах, че не съм направил комуникацията коректно защото и двете поведение търсят едно и също conversationId и май има шанс да се прецака комуникацията. Логиката при съобщенията разменени при сприятеляване е подобна, но накрая се изпраща съобщение информиращо

администратора за новото приятелство, той от своя страна записва това в онтологията.

Другите моменти където гражданин комуникира с админа са в началото на хода където всеки гражданин желае да обнови информацията за околността си, при извършване на движение агентът информира координатора на кое място иска да се премести и при край на хода, за да покаже че е готов. Тези съобщение са прости INFORM или REQUEST съобщения. В отговор на съобщението за околността request-self-and-environment получава няколко съобщение носещи информация за хора, места и приятели.

Съответно що се отнася до администратора на града, то повечето му комуникация са съобщения за край на хода на определен агент, обявяване на нов ход за всички агенти и безкрайни отговори на запитванията за актуална информация на гражданите. Другите съобщения които получава са за записване на приятелства и на движенията на агентите.

Онтология в действие

При инициализиране на града, онтологията се изпълва с нови индивиди от класовете Person, Place и Job. Това се случва чрез OneShotBehaviour на администратора, който се извиква чрез GUI и с данни въведени от потребителя извиква три метода за генериране на хора, работи и места. Хората се генерират въз основа на бройката хора зададена от потребителя. Всичките им черти, външен вид и статистики са произволни. За всеки генериран човек както и за всички хора заредени от базата данни се създават места за живеене било то къщи или блокове и хората се разпределят по тях. Класът CityOntology работи с онтологията и чрез метода му addPersonToOntology всеки човек се зарежда като индивид в онтологията. Създават се аксиоми за всичките му data properties, добавя се мястото му на местоживеене и аксиомите за връзката му с мястото livesIn и owns където е приложимо. На този етап в онтологията съществуват индивиди за всички хора и връзките им с места, които все още не са заредени в онтологията.

Вторият метод създава индивиди за различните професии с цифри за нужните им умения и възнаграждението които са произволни в дадени лимити. След което се добавят в онтологията.

Третия метод е за създаване на останалите места в града. Той е последен защото трябва да знае колко жилищни сгради ще има в града. Използвайки числото зададено от потребителя за желаната бройка произволни сгради и известното число на жилищни сгради се избира най-малката n на n матрица която побира всички сгради като празните места се запълват с изоставени сгради. След това се добавят улици така че всички сгради да са достъпни от поне една улица. Получилата се карта от x на y брой елементи съдържаща всички сгради в града е завършената карта и текущата им позиция се записва като x и y координати. След като градът е „построен“ всички сгради се зареждат в онтологията.

По време на работа на симулацията след като града е бил създаден работим с онтологията когато се нуждаем от информация като „Къде е даден агент?“ „Кой е около него?“ „Кои са приятелите на човек А?“ „Кои хора са на позиция Б?“ „Дали място В има свободни позиция за Г?“(последният въпрос е възможен, но не е имплементиран в момента). Това е информация която се съдържа в онтологията под формата на object properties аксиомите isFriendsWith, isLocatedAt, worksAt, hasStaffSlots, isStaffedBy и други. По време на работа тези връзки между индивидите се създават, унищожават или променят.

Потребителски интерфейс

Потребителският интерфейс или GUI на приложението представлява JFrame десктоп приложение с минимален дизайн. Използва се card layout стил за представяне на информацията като има лява и дясна част като при различните изгледи ако на едната страна е представена информация за това което виждаме то от другата страна можем да манипулираме обекта чрез бутони или полета. В началото тъй като града още не е създаден имам само опция да го създадем, като за тази цел въвеждаме информация за желаните от нас хора и места в града. В този момент можем да видим и списък с хора записани в базата данни като ако искаме можем да ги добавим като индивиди в новия град. След създаване на града имам по-

голям избор от възможни действия. Ако отворим картата на града ще видим n на n „карта“ на града изградена от цветни квадратчета като всеки цвят символизира различно място било то улица, къща, полицейска станция или изоставена сграда. При натискане на някой квадрат ще се появи информация за конкретното място от рода на името на мястото, работни места, колко смени има, кои хора в момента са тук. Ако списъкът с хора не е празен то можем да кликнем върху дадено име за да получим повече информация за него подобно на информацията за място. Другите бутони на екрана са за началото на нов ход или добавянето на множество ходове под ред. Има и бутон за показване на всички хора в града. Когато сме на екрана с информацията на гражданин можем да го добавим в базата данни като му дадем ново име тъй като произволните хора са креативно наименувани Person1, Person2, Person3… При започване на ход симулацията ще прогресира, но все още това няма директна визуална индикация, тъй като самите хора не са изобразени на картата все още. Ако преминем през няколко хода и прегледаме агентите ще видим че са на други места, стресът им може да не е началният 0, може да са сформирали приятелства които можем да видим в списъка им с приятели в личната им информация.

Цялостната система

На този етап на проекта симулацията работи в много базов вид с минимални и „глупави“ интеракции между агентите, които даже не се изобразяват прилично за окото на потребителя, но ако пуснем agent sniffer ще видим лавина от съобщения между граждани и координатора и граждани и други граждани. Те комуникират правилно чрез ACL протокола но за цялата тази комуникация визуалните промени на този етап са минимални в най-добрия случай. Това обаче може да се нарече адекватна стартова позиция за създаването на по-сложен проект, понеже имаме главната работеща логика, а тя е именно комуникацията между агентите и боравенето с онотлогията.

Като бъдещо развитие на проекта смятам да подобря текущия проект като го разширя чрез добавянето на главен „герой“ и „злодей“ за да се добавят по0влиятелни агенти в симулацията. Също ще се подобри логиката за

вземане на решенията като вече ще гледа неща като текущото състояние на агента, характера му, хората около него ако е необходимо или приятелите/колегите му. Чрез задълбочаване на логиката зад решенията му и добавянето на нови действия като част от които са потенциално деструктивни смятам че това ще вдъхне много повече „живот“ в симулацията. Естествено за това трябва да се подобри изобразяването на града за по-лесно следене и дебъгване на симулацията. Крайната цел за бъдещия проект вероятно ще бъде да се добавят състояния за „победа“ на съществуващите агенти и агентите, които ще се добавят тогава. Тогава ще може да се добави логика за малко по-дългосрочно планиране с потенциални предателства и други деструктивни действия. Но за този бъдещ проект първо трябваше да се реализира текущия, за да се създаде основа върху която да се надгражда.