1. Volg de tutorial en omschrijf daarna in één paragraaf wat deze tool anders maakt dan andere programmeertalen, wat zijn de voor- en nadelen?

Netlogo is het meest gebruikte agent based modelling tool ter wereld. Het is het makkelijkst om te leren, en kan binnen 3 uur aangeleerd worden. De software is laag drempelig waardoor modellen makkelijk geimplementeerd kunnen worden. Zo kan er heel gemakkelijk knoppen/slides toegevoegd worden via de interface. De software bevat daarnaast al compelexere modellen zodat mensen deze kunnen bekijken en begrijpen. Universiteit, het nieuws en de media maken er gebruik van.

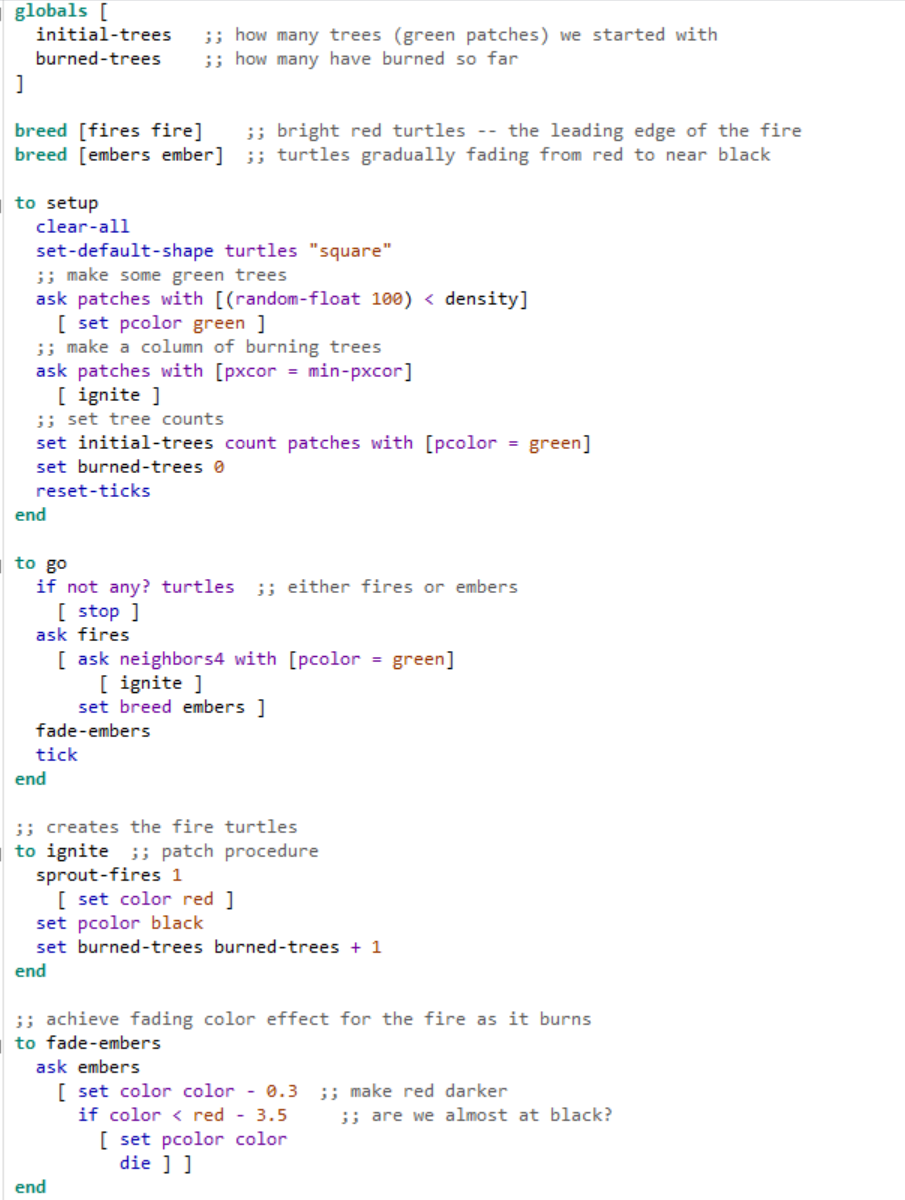
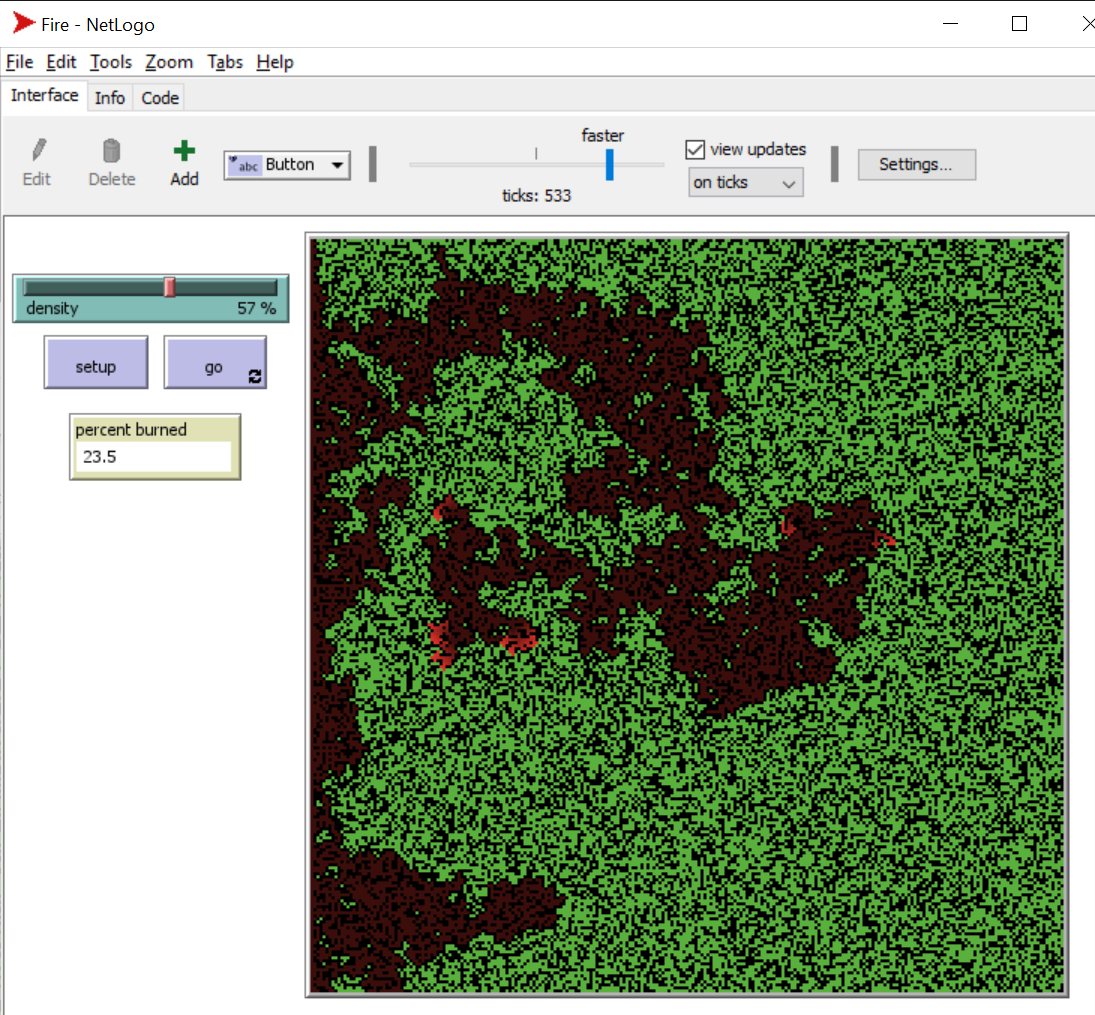
Daarnaast heeft Netlogo veel mogelijkheden. Het is uitdrukkelijk/expressief genoeg om high end complexe modellen te modelleren. Onderzoekers kunnen ook gemakkelijk modellen bewerken en publiceren. Delen van projecten gaat gemakkelijk, en daardoor kunnen modellen bevestigd/uitgedagen worden.

Ik heb voor Netlogo gekozen omdat het in mijn ogen het bekendst is bij iedereen, en daardoor er waarschijnlijk ook het meest over te vinden valt. Daarnaast is netlogo ook gebasseerd op Python, een taal waar ik al enige ervaring in heb. Ook sprak Netlogo mij aan omdat het laagdrempellig is.

Ik heb specifiek niet voor Unity gekozen. Unity is geschreven in C en C++, waar ik totaal geen ervaring mee heb. Ik wilde het mezelf dus niet moeilijker maken dan nodig. Ook vergt het veel ervaring om er mee te kunnen werken omdat het vrij complex is. Wel is het mogelijk om hele mooie programmas te schrijven in Unity.

Daarnaast hebben we Mesa. De keuze voor het kiezen van een tool was lastig, want ook mesa is op python gebasseerd. Ik heb Mesa niet gekozen omdat ik Netlogo toch duidelijker en bekender vond aanvoelen.

Als je een bestaand voorbeeld gebruikt (Netlogo heeft er vele): beschrijf de kernfunctionaliteit van het voorbeeld en voeg minimaal een element toe aan de omgeving en licht toe. (bijvoorbeeld: het meest eenvoudige economiemodel in Netlogo is agenten die random geld uitdelen, je kunt dat aanpassen naar geld uitdelen aan agenten die dichtbij hen staan.



1. We definieren een staat en drie functies waarmee we een stateful agent abstract omschrijven:
   * Een initiele staat i0∈I, waarbij ***I***alle mogelijk interne staten van de agent zijn
   * Een functie "See" of "Perceive", die een mapping maakt van elke staat in de omgeving tot een staat die de perceptie van de agent van de omgeving aangeeft. Dus: *See:* S⟶P, waar ***S*** de staat of serie van staten van de omgeving is en ***P*** de perceptie van die omgeving
   * Een functie "Act" die een perceptie van de omgeving neemt en een toepasselijke actie kiest. Dus *Act:* I⟶A, waar ***I*** een interne staat is en ***A*** een actie.
   * Een functie "Update" die een staat ***I*** neemt (soms ***D*** genoemd) en perceptie ***P***, en een nieuwe staat ***I***oplevert, dus *Update:* I×P⟶I  
     Dit is een korte samenvatting van wat er staat in: "An introduction to Multi-Agent Systems" chapter 1.1 up to 1.4.  Die paragrafen kun je [hier](https://canvas.hu.nl/courses/27428/files/2223184/download) vinden.  
     Natuurlijk worden deze functies stukken complexer wanneer je agenten complexer worden.  
       
     Bijvoorbeeld: We hebben een simulatie met bankjes bomen en agents. De agents hebben een interne staat en die bestaat onder andere uit of ze op een bankje zitten,  of onder een boom zitten; ze kunnen alleen andere agents zien als samen op een bankje, en dan willen ook nog eens praten. De initiele interne staat van de agent bestaat dus uit hun positie en of ze, al dan niet alleen, op een bank of onder een boom zitten of niet. De functie See neemt een staat van de omgeving (dit is dus anders dan de interne staat van de agent) en gooit een percept eruit (zitten ze nu, al dan niet alleen, op een bank of onder een boom?) Ze zien alleen agents als ze samen meer dan 1 tick op een bankje spenderen. Als de agents op een bankje zitten blijven ze daar een tijdje zitten, en als ze er dan nog een agent komt te zitten zien ze die wel en gaan ze praten. Oftewel, de Act functie kijkt naar de interne staat van de agent. Als die beschrijft dat ze minder dan x lang op het bankje zitten, blijven ze zitten en als er nog een agent zit gaan ze daarmee praten. Dit verhaal is niet helemaal compleet natuurlijk, maar dan hebben jullie een idee van waar je aan kan denken.  
       
     Beschrijf in eigen worden wat elk van de 4 concepten in het algemeen beschrijft EN wat het in jouw specifieke simulatie betekent.
2. Beschrijf je omgeving op basis van de dichotomiën die [hier](https://canvas.hu.nl/courses/27428/files/2223184/download)op pagina 6 beschreven staan, en licht toe (dus niet alleen termen opsommen):
   * Accessible vs inaccessible
   * Deterministic vs non-Deterministic (Stochastic)
   * Episodic vs non-episodic (Sequential)
   * Static vs Dynamic
   * Discrete vs continuous
3. Bedenk een voorbeeld waarbij minimaal 3 dichotomies precies tegenovergesteld zijn en beschrijf of het veranderen van je omgeving op deze manier wel of niet iets zou toevoegen aan je simulatie.  
   Bijvoorbeeld: in een verkeerssimulatie maakt het soms wel echt uit voor de resultaten van je simulatie of de omgeving accessible is, maar bij het Nägelschreckenberg-model niet. Probeer te bedenken wat het effect van dit soort keuzes wat betreft je omgeving is op wat je kan leren/wat voor resultaten er uit je simulatie komen