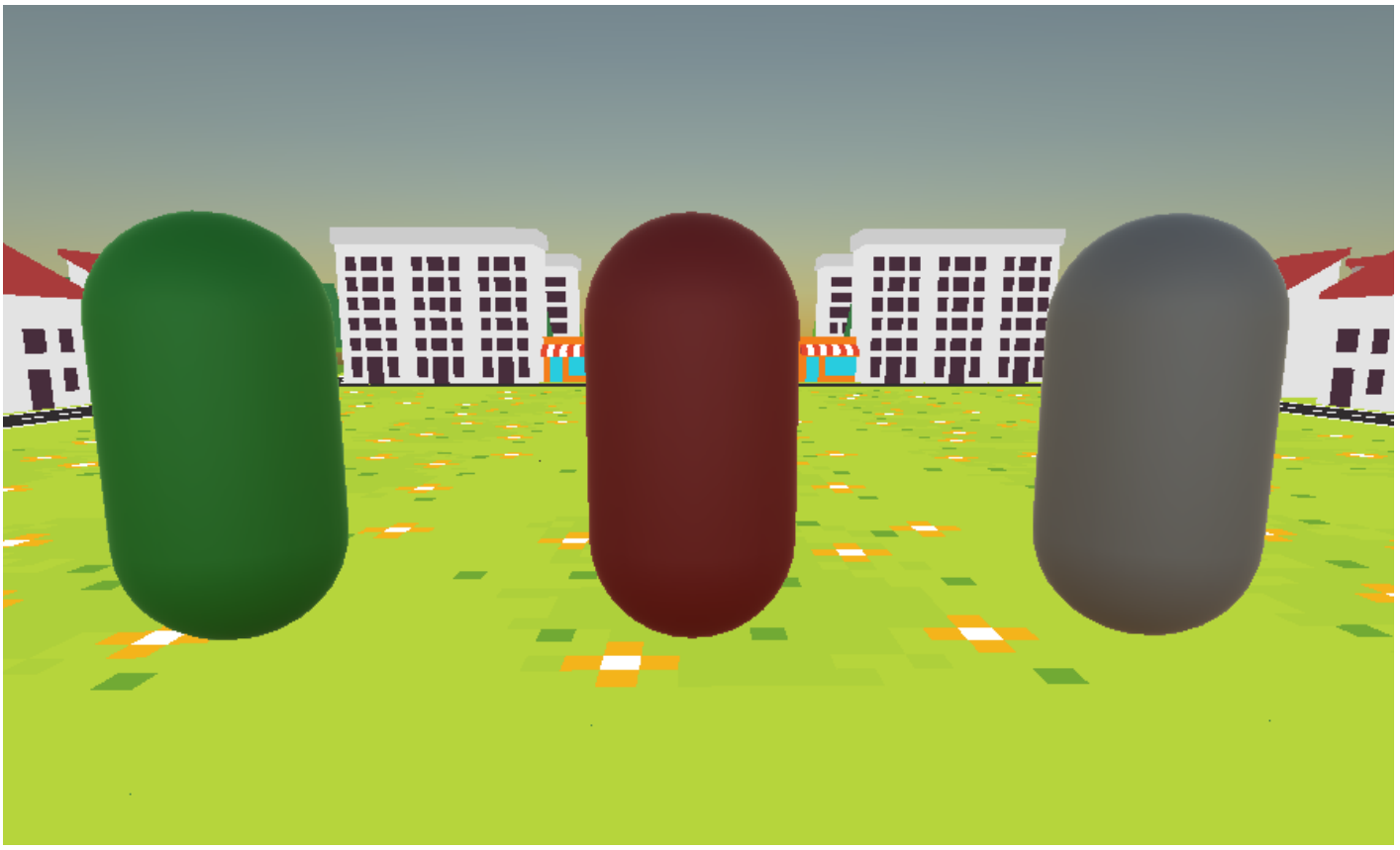
CHALMERS



# Utveckling av epidemisimulator

## Ett inlärningsverktyg för hur ett virus kan urarta sig

Examensarbete inom Data- och Informationsteknik

Johannes Lagergren
Jonatan Boman Karinen

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

EXAMENSARBETE

# Utveckling av epidemisimulator

## Ett inlärningsverktyg för hur ett virus kan urarta sig

Johannes Lagergren
Jonatan Boman Karinen

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2021

**Utveckling av en epidemisimulator**
Ett inlärningsverktyg för hur ett virus kan urarta sig.
Johannes Lagergren
Jonatan Boman Karinen

© Johannes Lagergren, Jonatan Boman Karinen, 2021

Handledare: Sakib Sistek, Chalmers Tekniska Högskola
Examinator: Peter Lundin, Chalmers Tekniska Högskola

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.
The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Omslagsbild:
De tre grupperna i en simulering, skapad av författarna

Institutionen för Data- och Informationsteknik
Göteborg 2021

# SAMMANFATTNING

På grund av rådande pandemi har ämnet virusspridning varit aktuellt i flera sammanhang. Det pågår just nu mängder med studier om hur man bör agera för att minska spridningen. Något vi och många andra har lagt märket till är att de flesta länder och dess invånare inte var förberedda för en situation som denna.
För att bidra till en framtida lösning utvecklades en mjukvara som nyttjar SIR-modellen, en matematisk modell inom epidemiologi som simulerar epidemier. Rapporten tar upp utvecklingen av simulatorn och tanken bakom våra val. Syftet med projektet var att utveckla ett inlärningsverktyg vilket kan användas i samarbete med undervisning för att förklara hur ett virus kan sprida sig genom en population samt vilka åtgärder gruppen kan ta för att försöka minska spridningen.

Simulatorn utvecklades med spelmotorn Unity och målet var att simulera en population i rörelse som blir utsatta av ett virus vilket användaren av simulationen själv kan konfigurera med inparametrar innan simulationen startar. I sin grundligaste form utvecklades en simulation som kan genomföra en epidemi i en kontrollerad miljö. Resultatet blev en simulation som uppfyller de mål som var satta men vi anser ändå att mjukvaran kräver mer utveckling, det går inte att säga hur väl simuleringen speglar ett verkligt scenario. Därför bör den färdiga produkten ej användas som bas för hur man bör agerar i verkligheten utan som en grund till vidareutveckling av liknande inlärningssystem.

**Nyckelord:** SIR-modell, epidemi, simulering, Unity,

# ABSTRACT

Due to the current pandemic the topic of how a virus spreads has been in the forefront of conversation. There are lots of studies going on about how one ought to act in order to decrease the rate of spread. We among others have noticed that most countries were not prepared for an event like this. To contribute with a future solution we developed a software that utilizes the SIR-model, a mathematical model in epidemiology which is used to simulate epidemics. This paper describes the development process and the reason behind our decisions. The purpose of this project was to develop a learning tool which can be used in combination with a lecture to aid in the learning of how a virus spreads through a population and what actions the group can take to slow the spread.

The simulation was developed with the game engine Unity and the goal was to simulate a population which get infected by a virus that the user of the application configures before the simulation is run. In its most basic form, a simulator which can implement an epidemic in a controlled environment was created. The result of the project is a simulator that meets the goals of the project, but we still believe that further development is necessary, it is not possible to say how well the simulation reflects a real scenario. Therefore, the final product should not be used as a basis for how to act during an epidemic rather as a base for further development of a similar learning system.



**Keywords:** SIR-model, epidemic, simulation, Unity

# FÖRORD

Vi vill ge ett stort tack till vår handledare Sakib Sistek. Du har hjälpt oss att gå från idé till slutprodukt. Genom din positiva energi, härliga samtal och vägledning har du hjälp oss att hålla huvudet högt och fokusera på uppdragen framför oss.

# TERMINOLOGI OCH FÖRKORTNINGAR

**Prefab** Prefabrication av ett objekt i Unity som kan instansieras vid senare tillfälle.

**API** Applikationsprogrammeringsgränssnitt (Application Program Interface).

**WYSIWYG** What You See Is What You Get, en princip som bygger på att produkten under utvecklingen hela tiden visas som om det vore den slutliga produkten.

# FIGURER

3.1 *Fågelperspektiv från simulationen som visar hur bostadshus samt arbetsplatser representeras i simulationen.* *8*

3.2 *Koden som tilldelar en icke-spelbar figur koordinater till ett hus.* *9*

3.3 *Enumen som håller de tre hälsostatusen.* *9*

3.4 *Koden som tar hanterar tiden i simulationen.* *10*

3.5 *Egenskaperna av en NavMesh Agent komponent.* *11*

3.6 *Funktionen SetDestination.* *11*

3.7 *En mottaglig icke-spelbar figur från gruppen [S] inom smittradien av en infekterad icke-spelbar figur i gruppen [I].* *13*

3.8 *Formel vilket räknar ut risken för en icke-spelbar figur att bli smittad per sekund då den befinner sig inom en smittradie.* *13*

3.9 *Design sketch för diagrammet.* *14*

3.10 *ProGrids arbetsmiljön i Unity.* *15*

4.1 *Simulatorns användarmeny.* *16*

4.2 *Spelkartan och knappar.* *17*

4.3 *Stapeldiagram över en simulering.* *18*

4.4 *Bild på tre Icke-spelbara figurer i grupperna susceptible, infectious och removed från vänster.* *18*

# Innehållsförteckning


Sammanfattning iii
Abstract iv
Förord v
Terminologi och Förkortningar vi
Figurer vii
1. Inledning 1
1.1 Bakgrund 1
1.2 Mål 1
1.3 Syfte 2
1.4 Frågeställningar 2
1.5 Avgränsningar 2
2. Teori 3
2.1 SIR-modell 3
2.2 Matematiken bakom SIR-modellen 4
2.3 Basal reproduktionskvot 5
2.4 Hur människor spenderar sin dag 6
2.5 Antal personer i hushåll 6
3. Metod 7
3.1 Verktyg 7
3.1.1 Unity 7
3.1.2 Visual Studio 7
3.1.3 GitHub och GitHub Desktop 8
3.2 Simulation utveckling 8
3.2.1 Implementation av icke-spelbara figurer 8
3.2.2 Implementation av en dag i simulationen 9
3.2.3 Icke-spelbara figurers rörelse 11
3.2.4 Implementation av SIR-modellen för virusspridning 12
3.2.5 Utveckling av diagram för visualisering av virusspridningen över tid 14
3.2.6 Konstruktion av spelkarta och kamera 15
4. Resultat 16
4.1 Meny 16
4.2 Spelkartan och Kameran 17
4.3 Stapeldiagram 17
4.4 Icke-spelbara figurer 18
4.5 SIR-modellen 19
5. Diskussion 20

5.1 Varför denna version av SIR-modellen? 21
5.2 Användning av matematiska modeller för utveckling av visuella verktyg 22
5.3 Etik kring simuleringar 22
6. Slutsats och framtida rekommendationer 23
Källor: 24
Appendix: 26

# 1. Inledning

Detta inledande kapitel kommer ta upp bakgrunden till varför projektet utförs samt vad syftet och målet med det är. Fortsättningsvis introduceras de frågeställningar som ska diskuteras samt de avgränsningar som satts för projekt.

## 1.1 Bakgrund

Under den pågående pandemi cirkuleras mycket information samt missinformation. Myndigheter har förmedlat hur en bör agera för att minimera spridningen av SARS-CoV-2, trots detta är det många som väljer att inte följa råden. Vi tror att en möjlig bidragande faktor till varför folk väljer att inte följa råden kan vara att det är komplicerat att kvantifiera och visualisera hur ens handlingar påverkar spridningen av ett virus, speciellt om man inte är insatt i ämnet. Detta projekt är genomfört på Chalmers tekniska högskola och går ut på att försöka utveckla en mjukvara med syfte att hjälpa inlärningen om hur ett virus kan urarta sig genom att visualisera spridningen i en simulation.

Simulationer ger möjligheten att utreda flera scenarier med identiska utgångspunkter, där enstaka variabler skiljer sig åt. Detta ger möjligheten att observera skillnaden i spridningen av ett virus beroende på enstaka variabler som modifierats. Man kan öka förståelsen om hur virus kan spridas genom att presentera resultaten av en spridning med visuella hjälpmedel i en simulation istället för matematiska tabeller i ett dokument.

## 1.2 Mål

Målet med projektet är att utveckla en mjukvara som simulerar simplifierade scenarion där diverse luftburna virus kan initieras av användarens inparametrar. Mjukvaran ska också, med hjälp av grafer och visuella verktyg, förmedla hur viruset spridit sig genom populationen. Mer konkreta mål är:

- Implementation av en meny där användaren kan välja inparametrar för simulationen.
- Skapa en spelkarta som simulationen utspelar sig på, där icke-spelbara figurer kan navigera och interagerar med omvärlden.
- Icke-spelbara figurer som speglar människor i rörelse med egenskaperna att bli smittade, smitta andra och återhämta sig.
- En graf som visuellt förmedlar hur ett virus kan urarta sig och sprids genom att kontinuerligt presentera antalet mottagliga, smittade samt återhämtade.
- Implementation av virusspridning vilket följer en SIR-modell.

## 1.3 Syfte

Syftet med projektet var att skapa en mjukvara som, med hjälp av matematiska modeller inom epidemiologi, simulerar och använder visuella hjälpmedel för att förmedla hur ett virus kan urarta sig, samt hur människors handlingar påverkar spridningen. Förhoppningen med verktyget är att stärka uppfattningen om hur en smittspridning kan gå till, alternativt komplettera den kompetens som redan finns.

## 1.4 Frågeställningar

Projektet kommer lyfta fram frågeställningarna:

Varför utveckla visuella representationer för de matematiska modellernas resultat istället för att visa matematiska beräkningar och tabeller?

Vilken version av SIR-modellen kan användas då man utvecklar ett inlärningsverktyg för hur virus sprider sig?

## 1.5 Avgränsningar

För att projektet inte ska bli för brett så har vissa avgränsningar tagits i beaktning:

- Simulationen är avsedd att spegla en epidemi istället för en pandemi. Detta är eftersom epidemier är av mindre skala samt lokala i sin geografiska utsträckning.
- Programmet anses vara ett demo för ett inlärningsverktyg om hur ett virus kan urarta och sprida sig i början och av en epidemi och inte en simulation som ska göra några förutsägelser om pågående pandemi.
- Tillverkning av grafik är tidskrävande och kommer ej göras av oss i detta projekt. Därför används färdiga sprites och texturer i form av betal-vara.

Målet är absolut inte att skapa en mjukvara som på något sätt ska användas för att förutsäga vad som kommer hända i verkligheten eller hur en person ska agera. Programvaran ska endast användas som ett komplement till övrig information och kompetens kring smittspridning.

# 2. Teori

Kapitlet introducerar den matematiska modell som används i simulationen samt teorin och matematiken bakom den. Fortsättningsvis introduceras statistiken som används för att bestämma hur icke-spelbara figurer ska spendera sina dagar i simulationen. Kapitlet avslutas med att ta upp antalet icke-spelbara figurer per hushåll.

## 2.1 SIR-modell

Fackmodeller existerar för att simplifiera de matematiska modeller som används för att räkna ut hur en infektionssjukdom sprids inom en population beroende på tid. Inom vår simulation har vi valt att använda SIR-modellen. SIR-modellen är en fackmodell som publicerades första gången år 1927 av William Kermack och Anderson McKendrick i “Contribution to the Mathematical Theory of Epidemics” [1]. Där beskriver de tre olika fack människor kan befinna sig i under en epidemi, [S]- susceptible, [I]- infectious, [R]- recovered. Dessa tre grupperna är grunden till SIR-modellen.

- Susceptible - Är gruppen individer som är mottagliga till ett virus. Detta inkluderar alla som är mottagliga, även individer som har varit immuna tidigare men nu återigen mottagliga.
- Infected - Är gruppen individer med tillräckligt hög koncentration av virus för att föra det vidare. Individer kan vara smittade av ett virus fast ej ha tillräckligt hög koncentration för att föra viruset vidare.
- Recovered- Är gruppen individer som på något sätt är immuna. Det inkluderar personer som tidigare varit infekterade och tillfrisknat, personer som är naturligt immuna eller till och med personer som är döda.

Slutligen bör man veta att det finns ett flertal olika variationer av SIR-modellen. Versionen som vi använder oss utav tar ej hänsyn till demografi, vilket betyder att populationsstorleken är konstant samt att faktorer som ålder och kön ej tas till hänsyn. Befolkningen är också homogen vilket betyder att alla har samma risk att bli smittade i fall att de befinner sig inom en riskzon, vilket inte är fallet i verkligheten. Anledningen till varför vi valt att använda oss av denna version av SIR-modellen, som ej tar hänsyn till demografi, är för att spannet av vår simulation ej kommer att vara tillräckligt långt för att det ska vara nödvändigt att introducera nya individer till befolkningen.

## 2.2 Matematiken bakom SIR-modellen

Innan matematiken introduceras är det viktigt att nämna vilka antagande som görs vid användning av denna version av SIR-modellen:

- Populationen är konstant, vilket betyder att det inte föds några nya personer samt att det inte finns några personer som migrerar.
- När någon återhämtar sig ger det dem total immunitet.
- Det finns ett fast antal mottagliga personer som kan bli infekterade.
- Det tar ett fast antal dagar att återhämta sig när man har blivit smittad.

Dessa antaganden görs alltså vid användning av SIR-modellen. Ytterligare nödvändig information för att förstå matematiken är:
β - är spridningshastigheten per capita, antalet personer en individ kommer i kontakt med och smittar per dag.
γ - är 1/x där x är antalet dagar det tar för individer att tillfriskna efter att de blivit infekterade. γär då hur mycket friskare en person blir per dag.

Antalet personer i fack [S] kan bara minska. Antalet personer i fack [I] ökar då [S] minskar och minskar då fack [R] ökar. Fack [R] kan bara öka. För att indikera övergången från ett fack till ett annat används följande notation:

$$S \rightarrow I \rightarrow R$$

SIR-modellen använder tre differentialekvationer för att beräkna ändringen i [S]-, [I]- och [R]-facken över tid:

$$dS/dt = -\beta SI \quad (1)$$

Minskningen av individer i grupp [S] är beroende av tidsenheten (t) men också beroende av hur många individer som redan befinner sig i grupp [S] eftersom ju större [S] är desto högre är(β), risken att en individ från grupp [I] kommer i kontakt med en person från grupp [S].

$$dI/dt = \beta SI - \gamma I \quad (2)$$

Antalet individer i grupp [I] ökar i takt med minskningen av individer i grupp [S], där av positiv $\beta SI$. Minskningen sker i takt med att individer blir friska och hamnar i gruppen [R], där av $-\ \gamma I$ (tiden det tar att tillfriskna multiplicerat med antal personer i fack [I]).

$$dR/dt = \gamma I \quad (3)$$

Antalet individer i grupp [R] ökar i takt med minskningen av individer i grupp [I], där av $\gamma I$ . [2]

## 2.3 Basal reproduktionskvot

$R_0$, även kallad basal reproduktionskvot, benämns frekvent som ett av det viktigaste värdena när det kommer till epidemiologi. $R_0$kan definieras som det genomsnittliga antalet personer smittade efter att en infekterad person placeras in i en population full av mottagliga personer [S].

Detta betyder att om

$$R_0 < 1$$

kommer viruset inte att spridas tillräckligt fort vilket leder till att viruset dör ut övertid. Tiden det tar för viruset att dö ut beror på hur lågt $R_0$är. Då

$$R_0 > 1$$

betyder det att viruset sprider sig tillräckligt fort för att det skulle finnas en möjlighet att viruset sprider sig till resten av populationen. Detta betyder också att generellt så gäller det att ju större $R_0$är desto större är risken att viruset sprider sig i större utsträckning.

För att få fram $R_0$använder vi oss utav β och γ:

Om γ är hur mycket friskare en person blir per dag, alltså $\gamma = 1/x$ där x är antalet dagar det tar för att bli frisk. Så kan man säga att

$$1/\gamma = x\ antal\ dagar\ det\ tar\ att\ bli\ frisk.$$

Anta att den första och enda infekterade individen i grupp [I] rör sig som vanligt och kommer i kontakt med den resterande mottagliga populationen i grupp [S]. Detta kan då leda till att den infekterade personen sprider vidare viruset med en spridningshastighet β under infektions perioden $1/\gamma$. Alltså kan man anta att den första infekterade personen kommer infektera

$$R_0 = \beta/\gamma$$

personer. [2]

## 2.4 Hur människor spenderar sin dag

Då man utvecklar en simulation med syfte att efterlikna människor krävs en grundförståelse om hur människor i grupp rör sig dagligen. Det är därför viktigt att basera besluten om människors rörelse och handlingar i simulationen på verklighetsbaserad data. Our World In Data (OWID) är en forskningsgrupp, baserad i Oxfords Universitet, som arbetar med globala problem. Gruppen har tagit fram en studie som sammanfattar data om hur människor i ett antal länder spenderar sina dagar [3]. Studien används i projektet som en bas för att få ett verklighetsbaserat beslut om hur de icke-spelbara figurerna i simulationen ska röra sig.

Enligt OWID så spenderar den genomsnittliga individen sin dag på följande vis:

- Betalt arbete - 149-315 minuter
- Sover - 480-540 minuter
- Fritid - återstående tiden av dygnet, 585-811 minuter

En detalj som bör tas i beaktning, vilket de lyfter fram i studien, är att deltagarna i studien är mellan åldrarna 15-69. Detta innebär att även arbetslösa personer är inräknade i statistiken, vilket leder till att antalet timmar av betalt arbete sänks.

## 2.5 Antal personer i hushåll

Som vi ser i tidigare delkapitel 2.4 spenderar man väldigt stor del av dagen i hemmet. Därför är det viktigt att basera storleken av hushållen i simulationen på realistiska levnadssituationer. Speciellt eftersom virusspridningen är i fokus och antalet mänskliga interaktioner spelar roll.

PRB (Population Reference Bureau) är en ideell organisation som specialiserar sig på att inhämta data om populationer. Enligt PRB är den genomsnittliga hushållsstorleken för alla kontinenter, förutom Asien då det saknas tillräckliga data, 3.26 personer [4]. Vi nyttjar därför statistiken från PRB vid distribution av antalet icke-spelbara figurer per hem för att spegla verkligheten.

# 3. Metod

Kapitlet introducerar verktygen och metoderna vi använt för att utveckla mjukvaran. Först går vi igenom vilka verktyg vi använt sedan redovisar vi utvecklingsprocessen för simulationen.

## 3.1 Verktyg

I följande avsnitt beskrivs de tekniska verktyg som användes vid skapandet av simulationen och motiveringen till varför de används istället för andra verktyg.

### 3.1.1 Unity

Unity är en utvecklingsplattform och spelmotor skapat av Unity Technologies. Utvecklingen i spelmotorn sker med hjälp av ett skript API baserat på programmeringsspråket C# och används primärt till utveckling av två- och tredimensionella datorspel [5]. Unity erbjuder användare av spelmotorn funktioner i form av rendering, skräpinsamling, användning av förfabrikerade objekt (prefabs), inbyggd kollisionsdetektering och en grafisk redigerare som implementerar WYSIWYG (What You See Is What you Get) principen. WYSIWYG underlättar utveckling eftersom den ger en direkt respons om hur produkten ser ut i nuläget.

Eftersom projektet har ett fokus på att visualisera en smittspridning var det viktigt att hitta en utvecklingsplattform som underlättade processen att rendera och visualisera spelobjekt, vilket Unity gör. En alternativ ledande spelmotor som erbjuder liknande funktionalitet är Unreal Engine [6]. Anledningen till att Unity valdes i slutändan är på grund av likheterna mellan C# och Java, till skillnad från Unreal som använder C++ vilket gruppen inte har lika mycket erfarenhet av.

### 3.1.2 Visual Studio

Visual Studio är en fullfjädrad IDE som erbjuder textredigering, kompilering, inbyggd git-funktionalitet, stöd för utveckling mot cloud-applikationer och även stöd för utveckling mot både mobil- och skrivbordsbaserade applikationer [7]. Visual Studio följer med i installationen av Unity och används till att redigera skriptfilerna för att manipulera objekten i Unity. Då Unity sköter kompileringen används Visual Studio bara som ett textredigeringsprogram i detta projekt.

### 3.1.3 GitHub och GitHub Desktop

GitHub tillämpar versionshanteringssystemet Git och är ett webbaserat versionshanteringsverktyg som lagrar alla versioner av ett program [8]. Användningen av GitHub tillåter oss att arbeta parallellt med mindre delar av programmet, i en skyddad miljö, för att sedan kunna sammanfoga de mindre grenarna med huvudgrenen. Detta leder till att huvudgrenen alltid kan erhålla en fungerande version av applikationen samt ger det oss möjligheten att hantera buggar i en skyddad och kontrollerad miljö.

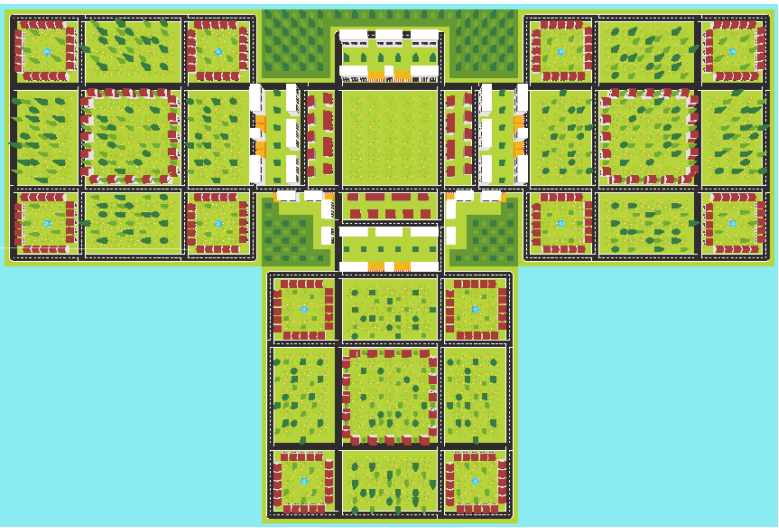
GitHub Desktop är ett verktyg som tillåter oss att interagera med GitHub utanför webbläsaren [9]. Primära anledningen till varför projektet använder sig av skrivbordsapplikationen är för att det grafiska gränssnittet underlättar versionshanteringen. Applikationen underlättar även processen att växla mellan versionsgrenar vilket gynnar arbetsflödet.

## 3.2 Utveckling av simulatorn

I detta delkapitel går vi igenom utvecklingen av de viktigaste delarna i simulationen, hur de implementerades och den generella strukturen av komponenterna kommer tas upp.

### 3.2.1 Implementation av icke-spelbara figurer

Människorna i simulationen representeras av icke-spelbara figurer. Vid initiering av simulationen instansieras varje icke-spelbar figur utifrån en prefab, en prefabricerad mall av en icke-spelbar figur. Därefter blir de tilldelade ett hus, ett jobb och en hälsostatus. Husen och arbetsplatserna är koordinater på spelkartan som sparas hos varje enskild icke-spelbar figur. I *figur (3.1)* kan man se den visuella representationen av dessa koordinater: röda hus representerar bostadshus och vita höghus representerar arbetsplatser.



*Figur 3.1 Fågelperspektiv från simulationen som visar hur bostadshus samt arbetsplatser representeras i simulationen.*

Icke-spelbara figurers syfte i simulationen är att återspegla människor. Det är därför ytterst viktigt att implementera verklighetsbaserade beteenden. Detta uppnås genom att tilldela hushåll med statistiken från kapitel 2.4 i åtanke. Studien redovisar att det genomsnittligen bor 3 personer i ett hushåll, koden i *figur 3.2* ser till att antalet icke-spelbara figurer i ett hus stannar runt genomsnittet [4].

```
home = GameObjectLists.instance.homes[index % sizeOfHomes].transform.position;
```

*Figur 3.2. Koden som tilldelar en icke-spelbar figur koordinater till ett hus.*

Arbetsplatserna tilldelas på liknande sätt och strävar efter ett maximum på 10 personer per arbetsplats. I kapitel 3.2.2 redovisas hur icke-spelbara figurers dagar återspeglar den genomsnittliga personens dag med hjälp av studien från kapitel 2.5.

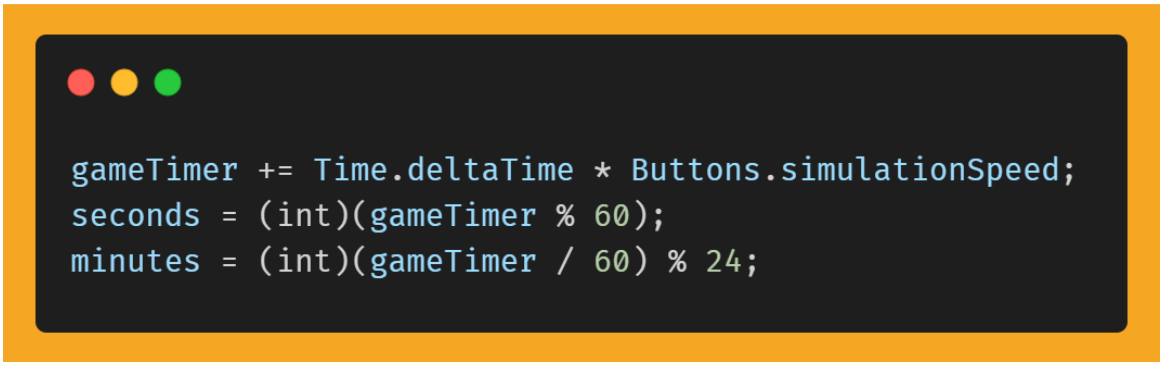
En icke-spelbar figurs hälsostatus definieras med hjälp av en enum. Enums är en datatyp som består av en definierad mängd konstanter. I detta fall innehåller enumen susceptible, infected och recovered, vilket är de tre hälsotillstånd som en icke-spelbar figur kan erhålla.

```
public enum InfectionState
    {
        Susceptible,
        Infected,
        Removed
    }
```

*Figur 3.3. Enumen som håller de tre hälsostatusen.*

### 3.2.2 Implementation av en dag i simulationen

Timmarna och dagarna inom simulationen hanteras av skriptet TimeHandler. Skriptet initieras då simulationen startas och mäter varje sekund med följande kod:



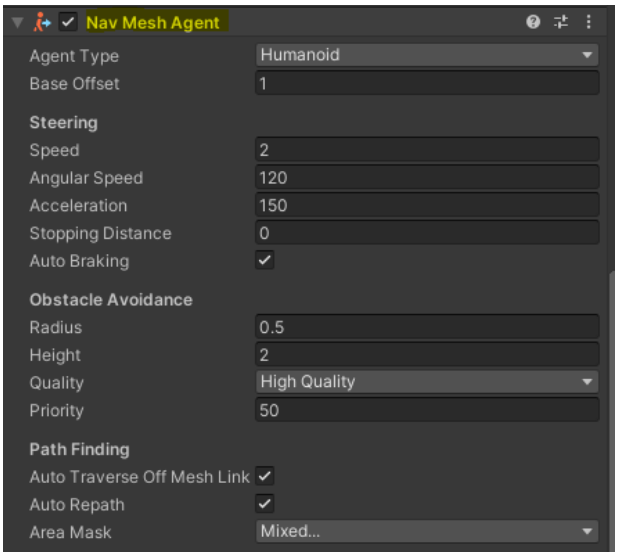

*Figur 3.4. Koden som tar hanterar tiden i simulationen.*

gameTimer är variabeln som beräknar antalet sekunder som har förflutit i simulationen. Detta görs genom att multiplicera Time.deltaTime med en variabel simulationSpeed. Funktionen deltaTime är en Unity funktion som returnerar 1 varje sekund i programmet. Variabeln simulationSpeed används ifall simulationen ska snabbspolas och kan anta värdet 1 eller 10 beroende på om användaren av simulatorn valt att aktivera snabbspolning.

Varje dag i simulationen är 24 minuter vilket speglar de 24 timmarna på en verklig dag. Varje enskild icke-spelbar figur spenderar sin dag enligt studien från Our World in Data på följande sätt:  8 minuter arbete, 9 minuter i hemmet och resterande 7 minuter är fritid. Då studien också tog hänsyn till arbetslösa och yngre personer så implementera vi en chans för de icke-spelbara figurerna att inte gå till jobbet. Det finns alltså en 33% chans att en icke-spelbar figur väljer att vandra runt i simulationen istället för att gå till sin arbetsplats. Detta leder till ett mer verklighetsbaserat beteende då inte alla personer går till jobbet samtidigt.  De 9 minuter i hemmet representerar 8 timmar sömn och 1 timme på morgonen att göra sig redo. Resterande 7 minuter fritid kan en icke-spelbar figur välja att vandra runt i simulationen eller stanna hemma.

### 3.2.3 Icke-spelbara figurers rörelse

Icke-spelbara figurer i simulationen utnyttjar Unitys inbyggda navigationssystem, NavMesh (förkortning på Navigation Mesh eller navigationsnät på svenska). Navigationsnätet är en datastruktur som initieras ovan på spelkartan innan runtime och memorerar de spelobjekt som kan navigeras över av andra spelobjekt med behörighet. NavMesh Agents (Navigationsnäts Agenter) är spelobjekt med komponenten NavMesh Agent (*se figur 3.5)*, vilket betyder att de uppnår kraven för att röra sig på navigationsnätet. Alla icke-spelbara figurer är navigationsnäts agenter.




*Figur 3.5. Egenskaperna av en NavMesh Agent komponent.*

I de två tidigare subkapitlerna 3.2.1 & 3.2.2 har vi gått igenom TimeHandler och hur en icke-spelbar figur blir tilldelad hus och arbetsplats. Dessa används sedan för att bestämma destinationen åt en NavMesh Agent. TimeHandler håller koll på när det är tid för en icke-spelbar figur att byta destination medan koordinaterna för deras hem eller arbetsplats skickas in i en funktion SetDestination (*se figur 3.6*) vilket uppdaterar koordinaterna till deras nya destination. När destinationen är uppdaterad använder agenterna navigationsnätet för att röra sig till destinationen.

```
private void SetDestination(Vector3 destination)
    {
        if (destination ≠ null)
        {
            Vector3 targetVector = destination;
            _navMeshAgent.SetDestination(targetVector);
        }
    }
```

*Figur 3.6. Funktionen SetDestination.*

### 3.2.4 Implementation av SIR-modellen för virusspridning

För att simulera en epidemi implementeras en SIR-modell. Detta betyder att en icke-spelbar figur kan tillhöra en av tre grupper: [S]- susceptible, [I]- infectious, [R]- recovered. Dessa tre grupperna är grunden till SIR-modellen:

- Susceptible - Är gruppen icke-spelbara figurer som är mottagliga till ett virus.
- Infected - Är gruppen icke-spelbara figurer som är infekterade av ett virus och kan föra det vidare samt återhämta sig.
- Recovered - Är gruppen icke-spelbara figurer som återhämtat sig och nu är immuna.

I början av simulationen tillhör varje icke-spelbar figur gruppen [S], förutom den mängd användaren har valt att instansiera som infekterade, dessa tillhör gruppen [I]. Under simulationens gång kan de från gruppen [S] bli infekterade och övergå till gruppen [I], ytterligare kan de från gruppen [I] återhämta sig och flyttas till gruppen [R].

Grupperna [S], [I], [R] ändras beroende på två variabler, den första är tiden (t). Alla grupper är påverkade av tiden på samma sätt. De från grupp [S] hamnar i fler situationer där de riskerar att bli smittade ju längre tiden går, de från grupp [I] riskerar att smitta fler ju längre tiden går och de från grupp [R] har större chans att återhämta sig ju längre tiden går. Den andra variabeln som påverkar förändringshastigheten i en grupp är dess storlek:

- Minskning av grupp [S] beror på antalet icke-spelbara figurer som redan befinner sig i grupp [S] eftersom risken att komma i kontakt med en infekterad icke-spelbar figur från grupp [I] ökar då [S] är större.
- Ökningen av antalet icke-spelbara figurer i grupp [I] resoneras på samma sätt. Risken att smitta en icke-spelbar figur från grupp [S] ökar då grupp [I] är större. Grupp [I] ökar i takt med minskningen av grupp [S].
- Antalet icke-spelbara figurer i grupp [R] ökar proportionellt med minskningen av grupp [I].

En icke-spelbar figur i gruppen [I] sprider ett virus med hjälp av två funktioner. Den ena funktionen agerar hjälpfunktion och detekterar alla icke-spelbara figurer inom dess smittradie, figur 3.7 visar scenariot då en mottaglig icke-spelbar figur befinner sig inom smittradien av en smittsam icke-spelbar figur. Den andra metoden kontrollerar om varje icke-spelbar figur inom smittradien tillhör gruppen [S] och utför samtidigt en sannolikhetsberäkning vilket beräknar ifall den icke-spelbara figuren ska bli smittad. Ifall att det är en mottaglig icke-spelbar figur och beräkningen visar att den ska bli smittad placeras icke-spelbara figuren i en lista. Fortsättningsvis loopas listan igenom och hälsostatusen för varje icke-spelbar figur i listan ändras till infected vilket leder till att den övergår från gruppen [S] till gruppen [I].

*Figur 3.7. En mottaglig icke-spelbar figur från gruppen [S] inom smittradien av en infekterad icke-spelbar figur i gruppen [I].*

Risken att bli smittad beräknas genom formeln i figur 3.8. Variabeln InfectionRisk utgör risken att bli smittad vilket bestäms av användaren innan simulationen påbörjas.

```
float probability = 1f - Mathf.Pow(1f - InfectionRisk, Time.deltaTime/10);
```

*Figur 3.8. Formel vilket räknar ut risken för en icke-spelbar figur att bli smittad per sekund då den befinner sig inom en smittradie.*

En infekterad icke-spelbar figur återhämtar sig med hjälp av en funktion som tar användarens input. Innan simulationen börjar får användaren välja antalet simulationsdagar det ska ta att återhämta sig från viruset, en dag är 1440 sekunder. Detta representeras i vår kod av en variabel InfectionTimer. Funktionen minskar InfectionTimer med 1 varje sekund. När Infection timer når 0 flyttas den återhämtade icke-spelbara figuren från gruppen [I] till [R]. Detta leder till att den icke-spelbara figuren ej längre kan smitta andra samt att den själv inte kan bli smittad igen.

Slutligen bör man veta att det finns flertal olika variationer av SIR-modellen. Versionen som vi använder oss utav tar ej hänsyn till demografi, vilket betyder att populationsstorleken är konstant samt att faktorer som ålder och kön ej tas till hänsyn. Befolkningen är också homogen vilket betyder att alla har samma risk att bli smittade. Detta leder till att viruset alltid kommer dö ut eftersom det inte introduceras nya icke-spelbara figurer till grupp [S]. Viruset muteras heller inte tillräckligt för att de från grupp [R] ska bli mottagliga igen.

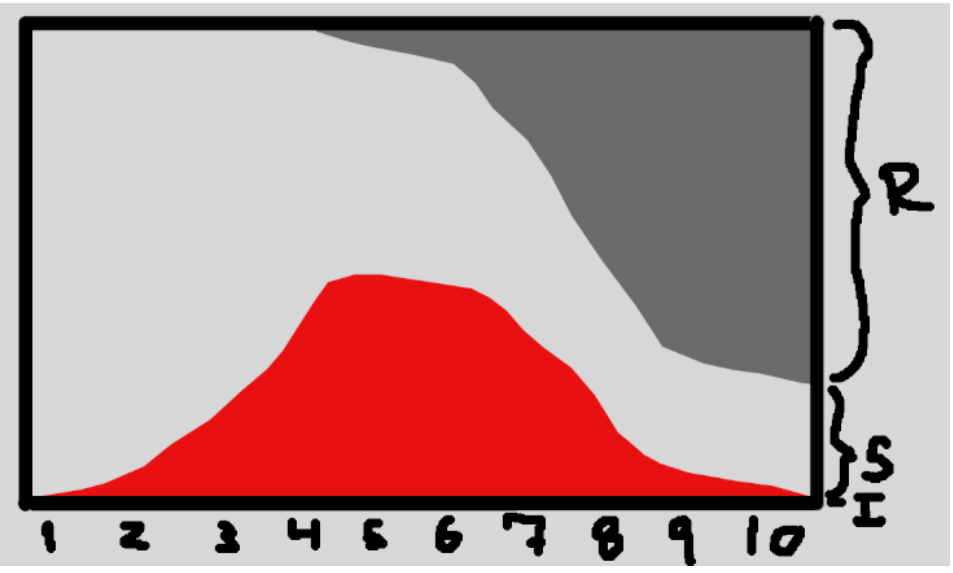
### 3.2.5 Utveckling av diagram för visualisering av virusspridningen över tid

För att kunna visualisera hur många icke-spelbara figurer som tillhör grupperna [S], [I] och [R] skapas ett stapeldiagram. Stapeldiagrammet färgläggs proportionellt till mängden icke-spelbara figurer i varje grupp, Figur 3.9 visar en design sketch för diagrammet.

För att illustrera stapeldiagrammet används Canvas, vilket är en klass i Unity som används för att rendera element ovanpå en scen [10]. Staplarna i diagrammet motsvarar antal icke-spelbara figurer på Y-axeln medan X-axeln motsvarar tid. Tiden använder simulationens dagar som enhet.

Under körningen av simulationen håller tre variabler koll på antalet mottagliga, infekterade och återhämtade icke-spelbara figurer. Dessa variabler ökar/minskar då en icke-spelbar figur byter grupp. Variablerna som motsvarar antalet icke-spelbara figurer i grupperna [I] och [R] avläses för att kunna rita diagrammet.

Ritandet av stapeldiagrammet sker med jämna tidsintervall genom att ta en ögonblicksbild av värdet i variablerna. Två staplar målas upp, en för värdet på antalet icke-spelbara figurer i gruppen [I] och en för [R]. Staplarna visualiseras med hjälp av Unitys vektorsystem och klassen RectTransform. För en intuitiv avläsning av diagrammet placeras [I] diagrammet i botten av grafen, alltså med startpunkt där Y är noll, och [R] längst upp i grafen med en startpunkt där Y är sitt maxvärde, se figur 3.9 för design sketch. Så länge inte grupperna [I] och [R] utgör alla icke-spelbara figurer i simulationen kommer det skapas ett tomrum mellan graferna. Detta tomrum är antalet i grupp [S].




*Figur 3.9. Design sketch för diagrammet.*

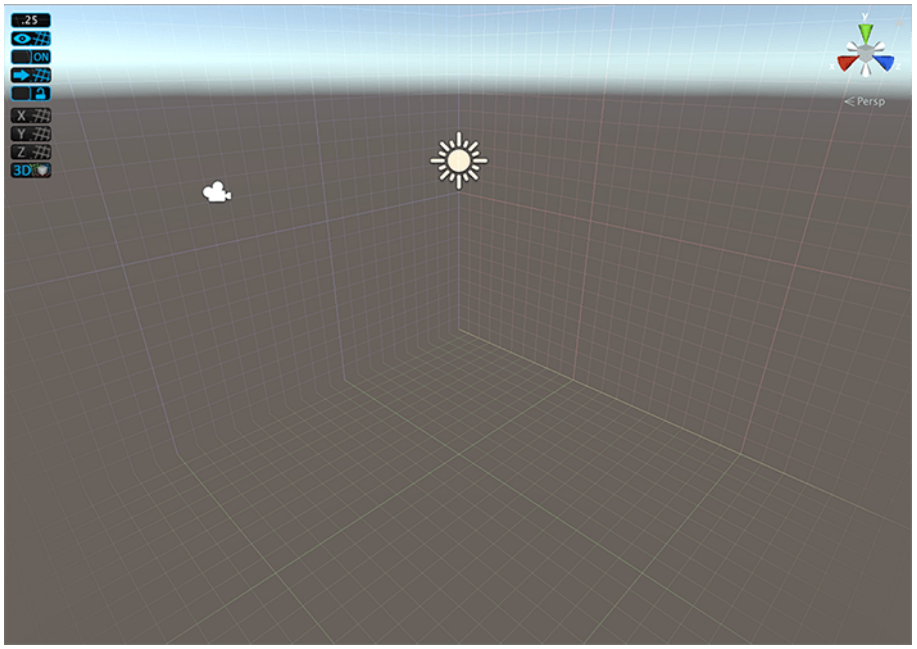
Vector2 är en struct i UnityEngine som ger en representation av 2D vektorer och punkter [11]. För varje stapel som målas ut skapas två Vector2-punkter, en för startpunkten och en för slutpunkten. Mellan start- och slutpunkten skapas ett spelobjekt med en

RectTransform komponent som gör att objektet blir en synlig geometrisk form i canvas:et [12]. Spelobjektet tar formen av en rektangel och får längden av avståndet mellan start- och slutpunkten.

Eftersom ett Unity canvas har en förinställd bredd kommer stapeldiagrammet efter en tid att fyllas ut. För att inte börja rita staplar utanför canvaset måste därför bredden mellan varje stapel ändras dynamiskt när det kommer in en ny. Skalningen av diagrammet sker genom att varje stapels nya position räknas ut av stapelns nuvarande index multiplicerat med kvoten mellan grafens fysiska bredd och antalet staplar. På det här viset förflyttas staplarna så att grafen visar alla staplar oberoende av hur länge simulering har kört.

### 3.2.6 Konstruktion av spelkarta och kamera

Spelkartan är en samling spelobjekt som tillsammans utgör en visuell representation av de olika koordinaterna i simulationen. Istället för att en icke-spelbar figur glider genom luften till en koordinat som representerar dess hem så ser man hur den förflyttar sig över en väg till ett faktiskt hus. Att skapa grafiska spelobjekt tar tid och då vårt mål var att arbeta med funktionaliteten i simulationen inhandlades ett färdigt paket med prefabs. För att underlätta uppbyggnaden av spelkartan användes Unity verktyget ProGrids. ProGrids implementerar ett rutnät av stödlinjer vilket prefabs kan fästas till.




*Figur 3.10. ProGrids arbetsmiljön i Unity.*

Unitys standard kamera är i förstapersonsperspektiv vilket betyder att kameran roterar över sin egen axel. För en simulering där du ska kunna observera allt uppifrån passar ett fågelperspektiv mycket bättre. Därför implementerade vi ett fågelperspektiv genom att skapa ett osynligt spelobjekt i mitten av kartan som kameran ska följa och rotera över. Detta leder till att kameran roterar runt spelkartan istället för sin egen axel.

# 4. Resultat

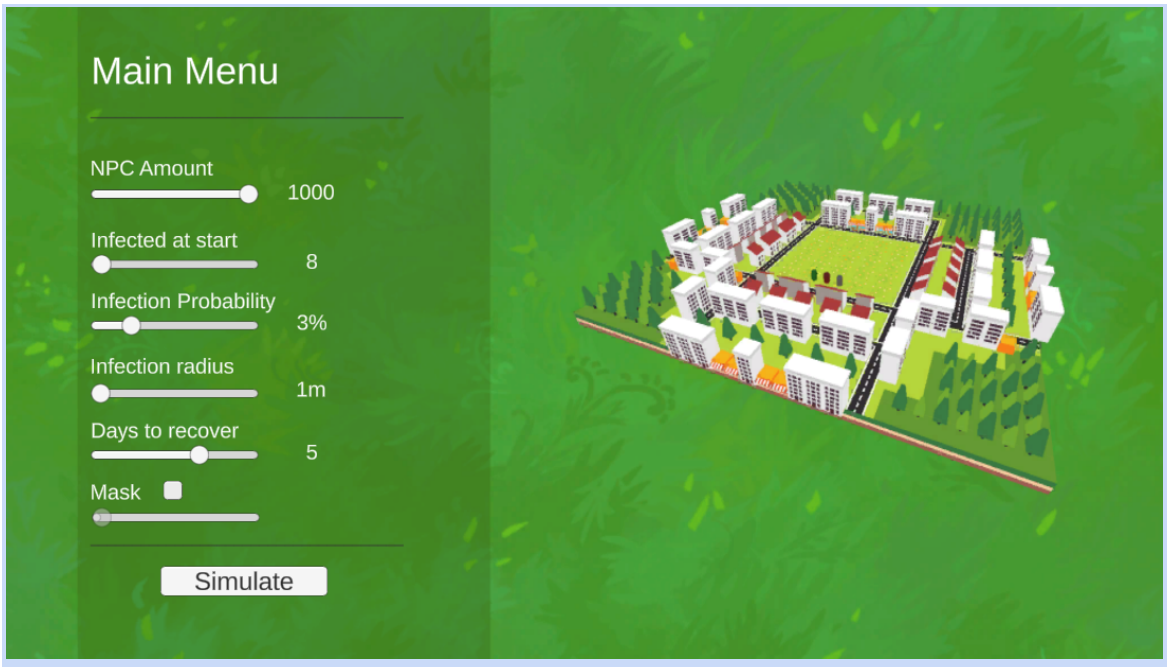
Resultatet från utvecklingen av simuleringens huvudkomponenterna presenteras i detta kapitel; menyn, spelkartan, kameran, grafen, implementation av icke-spelbara figurer och SIR-modellen tas upp.

## 4.1 Meny

Menyn är verktyget användaren kan nyttja för att styra simulatorns inparametrar. Huvudsakligen styrs parametrarna av sliders men det finns undantag i form av en kryssruta som har en på/av funktionalitet för att aktivera eller avaktivera en hel funktion, se figur 4.1. Variablerna användaren justerar berör främst icke-spelbara figurer och deras beteende samt viruset och dess egenskaper.

Parametrar som berör icke-spelbara figurer är antalet som initieras samt vilken andel av dessa som är smittsamma. Det finns även en kryssruta som bestämmer ifall icke-spelbara figurer ska bära mask samt en slider för procentantalet av populationen som gör detta.

Användaren kan även välja hur smittsamt viruset ska vara, detta är i form av en procent slider och är risken för en icke-spelbar figur att bli smittad då den befinner sig inom någons smittradie. Självaste smittradien kan också justeras samt antalet dagar det tar för en icke-spelbar figur att återhämta sig efter att den blivit infekterad.




*Figur 4.1. Simulatorns användarmeny.*

## 4.2 Spelkartan och Kameran

Spelkartan är ramen för simulationen och erbjuder en vy över icke-spelbara figurer i rörelse tillsammans med den omkringliggande miljön. Miljön består av prefabricerade spelobjekt som icke-spelbara figurer kan interagera med. Svarta vägar de kan gå på och röda samt vita hus vilket är destinationer som figurerna går mellan. Ovanpå kartvyn ligger även ett gränssnitt för knappar som kan användas för att visa statistik för simulationen, snabbspola simulationen eller återgå till meny skärmen.

För att underlätta navigationen av kartan skapades ett skript för att hantera kamerans rörelse. Kameran har ett fågelperspektiv över hela spelkartan (se figur 4.2) och kan styras med tangentbordet eller musen. Användaren kan zooma, rotera och förflytta vyn kameran har över simuleringen.

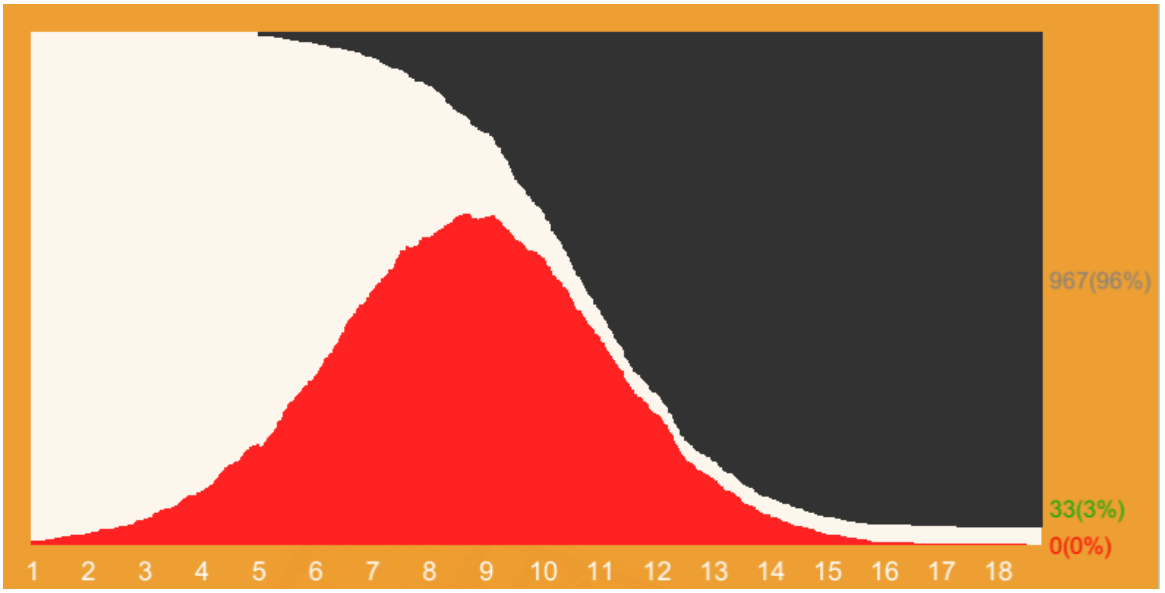
*Figur 4.2. Spelkartan och knappar.*

## 4.3 Stapeldiagram

Under körningen av simuleringen kan användaren klicka upp ett stapeldiagram vilket presenterar data från simuleringen genom att visualisera hur många icke-spelbara figurer som tillhör grupperna [S], [I] och [R] vid varje tidpunkt. I stapeldiagrammet nedan från figur 4.3 kan man se hur det nedersta röda diagrammet representerar antalet smittade, det översta mörkgråa diagrammet antalet återhämtade. Det vita området mellan diagrammen motsvarar antalet mottagliga. I appendix kan man se stegvis hur värdena i figuren förändrades i tre-dagars intervall.

Grafen uppdateras varje 30:e sekund i simulationen genom att göra en kontroll av storleken på varje grupp. Värdet för respektive grupp används sedan för att rita ut

staplarna och diagrammet bildas. Samtidigt uppdateras textfält till höger om stapeldiagrammet, dessa uppdateras också utifrån storleken av de olika grupperna och har i syfte att förmedla det exakta värdet av grupperna i heltal samt procent.

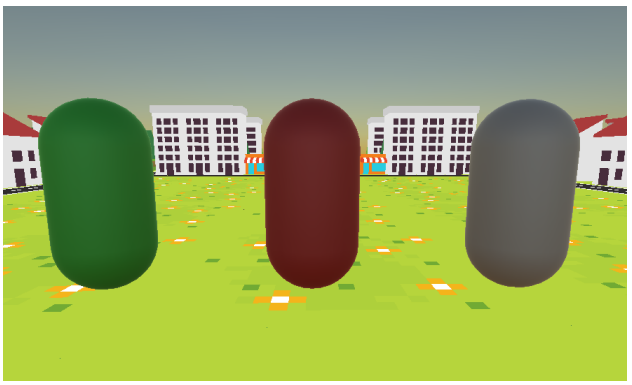



*Figur 4.3. Stapeldiagram över en simulering.*

## 4.4 Icke-spelbara figurer

Icke-spelbara figurer initieras utifrån prefabricerade spelobjekt från menyn. De initieras med en hälsostatus, koordinater för hem och arbetsplats samt en startdestination.

Hälsostatusen hos en icke-spelbar figur utgör vilken grupp den tillhör: [S], [I] eller [R]. Icke-spelbara figurer i gruppen [S] är av färgen grön och är mottaglig för ett virus. Gruppen [I] är av färgen röd och är smittförande. Gruppen [R] är av färgen grå och kan varken smitta eller bli smittad utav ett virus (se figur 4.4). Färgerna underlättar att identifiera vilken grupp en icke-spelbar figur tillhör men hjälper även till med visualiseringen av hur viruset sprider sig då övergången från en grupp till en annan blir tydligare.



*Figur 4.4. Bild på tre Icke-spelbara figurer i grupperna susceptible, infectious och recovered från vänster.*

Icke-spelbara figurers destination kan ändras vid olika tidpunkter under simuleringen. Detta leder till att de använder navigationsnätet för att röra sig mot den nya destinationen. Efter att de anlänt till sin destination står de stilla tills en ny destination anges.

Tidpunkterna då de olika destinationerna tilldelas styrs av tidsystemet Timehandler. För att efterlikna verkligheten tillbringar varje icke-spelbar figur sina nätter hemma, förmiddagar och lunchtider på sin arbetsplats och den övriga tiden på slumpvalt genererade platser inom ett avgränsat område. Ett visst antal icke-spelbara figurer rör sig också efter slumpvalt genererade platser under arbetstimmarna för att efterlikna arbetslösa i samhället. De sistnämnda är till för att efterlikna människor i rörelse i sociala platser som parker, torg och generella allmänna mötesplatser.

## 4.5 SIR-modellen

Resultatet från användningen av SIR-modellen är tre grupper [S], [I], [R] som förändras över tid. Icke-spelbara figurer kan övergå från grupp [S] till grupp [I] samt från grupp [I] till grupp [R], inte åt motsatt håll. Hastigheten av gruppens förändring beror på två faktorer, storleken av grupperna samt hur snabbt tiden går. Icke-spelbara figurer har olika egenskaper beroende på vilken grupp de tillhör.

- [S], susceptible - Är gruppen icke-spelbara figurer som är mottagliga till ett virus.
- [I], Infected - Är gruppen icke-spelbara figurer som är infekterade av ett virus och kan föra det vidare och återhämta sig.
- [R], Recovered - Är gruppen icke-spelbara figurer som återhämtat sig och nu är immuna.

Den implementerade versionen av SIR-modellen tar ej hänsyn till demografi. Detta betyder att de antal icke-spelbara figurer som initieras till grupp [S] är maxantalet mottagliga under simulationens gång. Effekten av att inte introducera nya individer till gruppen mottagliga är att antalet mottagliga tillslut kommer bli noll vilket leder till att viruset dör ut. En annan effekt av att inte ta hänsyn till demografin är att alla icke-spelbara figurer blir homogena. Detta leder till att alla har samma risk att bli smittade vilket inte är fallet i ett verkligt scenario.

# 5. Diskussion

Resultatet av simulationen uppfyller gruppens mål genom att tillfredsställa behoven kring funktionalitet och utseende. Menyn använder inparametrar för att specificera hur simulationen ska utspelas, spelkartan erbjuder en detaljerad bild över varje spelobjekt i simulationen, icke-spelbara figurer rör sig enligt verklighetsbaserad data och erhåller förmågan att bli smittade, smitta andra och återhämta sig, grafen visualiserar data angående viruset under simulationen och viruset sprids genom att följa en SIR-modell. I sin grundligaste form har det skapats en simulation som är utrustad med verktygen för att genomföra en epidemi i en kontrollerad miljö. Men genom vidareutveckling av funktioner och förbättringar av algoritmer kan simuleringen uppnå högre kvalité.

Antalet icke-spelbara figurer som initieras i simulationen just nu har en påverkan på resultatet. Vid max antal (1000) icke-spelbara figurer blir fördelningen av hus överensstämmande med antal personer som i genomsnitt befinner sig i ett hushåll (se avsnitt 2.4). Initieras ett betydligt lägre antal icke-spelbara figurer är detta ej fallet. Detta leder till att simuleringen kan spegla en helt annan verklighet. I dessa fall är spelkartan för stor för att simuleringen ska efterlikna en rimlig bild av en samhällssituation; det finns för mycket plats att röra sig på vilket sänker antalet “interaktioner” mellan icke-spelbara figurer till en nivå som inte längre speglar en samhällssituation.

För att lösa detta problem hade fler spelkartor i olika storlekar kunnat implementeras. Dessa kunde sedan initieras beroende på antalet icke-spelbara figurer så att även mindre populationer hade kunnat simuleras och fortfarande behålla en någorlunda rimlig samhällsbild, nu i en mindre miljö. Alternativt hade man kunnat implementera en funktion som autogenererar spelkartan beroende på antalet icke-spelbara figurer.

I vidareutveckling av programmet hade ett tillägg av fler säkerhetsåtgärder givit ett potentiellt mer verklighetsbaserat resultat. Eventuella åtgärder som kan implementeras är; självisolering, inkubationstid, social distansering och vaccin.

Smittade icke-spelbara figurer agerar just nu likadant som friska i simulationen, till skillnad från verkliga scenarion då personer kan stanna hemma efter insjuknande. Genom att implementera en självisoleringsfunktion som får icke-spelbara figurer att stanna hemma då det blivit smittade hade potentiellt simulerat en mer realistisk bild av en epidemi.

Inkubationstid innebär den tiden från det att man blir smittad till att man märker symtom, sjukdomen bryter ut. Inkubationstidensslut är ibland en indikator för att en individ nu kan smitta andra. I simulationen är det raka motsatsen. Eftersom vi inte implementerat någon inkubationstid börjar smittade icke-spelbara figurer smitta andra direkt efter de själva blivit smittade. För individens del har detta ingen betydelse men det kan leda till missvisningar i statistiken. Personer som plötsligt kan smitta andra direkt leder till att $R_0$ får ett högre värde än vad det egentligen bör. Genom att implementera en inkubationstid hade $R_0$ visats på ett mer korrekt sätt.

Även en implementation av social distansering skulle hjälpa resultatet eftersom det visuellt kan förmedla hur avstånd minskar smittspridning. I delkapitel 2.2 används variabeln β i SIR-modellen vilket är värdet på antalet personer en individ kommer i kontakt med och smittar per dag. Det betyder att viruset sprids långsammare då färre individer kommer i kontakt med en smittad person. Anledningen till varför vi inte har implementerat detta är på grund av hur vi hanterar icke-spelbara figurers rörelse. Genom att använda navigations-nät och agenter är det lätt implementera funktioner som rörelse men svårare att vidareutveckla att de ska hålla avstånd från varandra. Om vi istället hade utvecklat vår egen rörelse funktion från grunden hade social distansering gått smidigare att implementera.

Slutligen hade en vaccineringsfunktion kunnat implementeras genom att flytta en icke-spelbar figur från den mottagliga gruppen [S] till [R], vilket betyder att den inte längre kan bli smittad eller smitta andra. Om vi hade introducerat nya icke-spelbara figurer under simuleringens gång hade dessa kunnat ha en chans att "födas" vaccinerade, vilket sakta hade lätt till en vaccinerad population.

Resultatet som vi uppnått är tillräckligt men med lite vidareutveckling kan det förbättras. Det vi har utvecklat hittills bevisar att det är fullt möjligt att utveckla en epidemisimulator och fungerar som en grund för vidareutveckling.

## 5.1 Varför denna version av SIR-modellen?

Som tidigare nämnt finns det flera versioner av SIR-modellen. I avsnitt 2.1 nämns det att versionen som vi använder utesluter demografi. Detta innebär att populationsstorleken är konstant samt att alla icke-spelbara figurer är homogena, faktorer som ålder och kön har alltså ingen påverkan.

I ett verkligt scenario är det raka motsatsen. Enligt Jin Leng och Daniel R. Goldstein säger de att äldre personer är mottagligare till virusinfektioner och att det löper större risk för dödlighet [13]. Risken att bli smittad av ett virus samt tiden det tar att återhämta sig beror alltså på en faktor som ålder. I ett verkligt scenario är heller inte populationsstorleken konstant utan de introduceras hela tiden nya individer till smittpolen, vilket betyder att det tar längre tid för viruset att dö ut.

Varför använder vi då oss inte av en SIR-modell som tar hänsyn till demografiska faktorer? Eftersom vi endast observerar hur ett virus kan urarta sig, alltså hur det påbörjar och sprids inom en lokal population, så anser vi att det inte är av stor vikt. Inom vår tidsram runt 1-100 dagar hade det inte hunnit introduceras nya individer till smittpolen. Ifall att man utvecklar en simulation där man observerar ett virus och hur det påverkar en befolkning under längre sikt hade en SIR-modell med hänsyn på demografi varit mer aktuell.

## 5.2 Användning av matematiska modeller för utveckling av visuella verktyg

Resultatet av projekt bevisar att det är fullt möjligt att utveckla ett system som nyttjar matematiska modeller för att visualisera hur ett virus kan urarta sig och hur det sprids. Men varför inte bara använda den matematiska modellen för att redogöra detta?

Simulationsverktyget som utvecklats använder matematiska modeller som en grund för hur ett virus sprider sig. Vi använder oss av den matematiska SIR-modellen för att simulera en epidemi. Modellen nyttjar första ordningens differentialekvationer (även kända som ODEs). Resultat av en ODE kan vara svår att förstå sig på och konkretisera om man ej läst en högre nivå av matematik. Genom att istället visualisera resultatet i form av lättlästa grafer samt en simulerad population som visar övergången mellan grupperna [S], [I] och [R] tror vi att det kan leda till ökad förståelse om hur en virusspridning går till. Att de visuella aspekterna kan öka förståelsen är största anledningen till varför vi anser att en visuell representation av matematiska modeller kan användas istället för endast den matematiska modellen i sig självt.

## 5.3 Etik kring simuleringar

Simulationer kan köras hundratals gånger för att observera åtgärder vilket leder till minst spridning av ett virus samt minst smittrisk för en enskild individ. Trots detta är det fortfarande bara resultatet från en simulation som ej kan simulera en perfekt verklighet. Det som fungerar i simulationen behöver inte fungera utanför simulationen. Vad händer om en person följer simulations råd och utsätter sig för fara?

Det är viktigt att motverka detta genom att understryka resultatet från simulationen med att det inte är ett råd om hur man bör agera i en riktig epidemi. Resultatet är något att resonera och diskutera runt.  Vi tycker att man som utvecklare av ett sådant system har en skyldighet att tydliggöra till vilken utsträckning simulationen bör följas samt vilka bristerna den har. Vi har varit tydliga med att simulationen vi utvecklat i detta projekt är en grund till ett inlärningsverktyg samt ett komplement till den kompetens människor redan har idag.

# 6. Slutsats och framtida rekommendationer

Syftet med projektet var att utveckla en mjukvara vilket använder visuella hjälpmedel för att förmedla hur ett virus kan urarta sig. Resultatet visar på att det är fullt möjligt att utveckla visuella hjälpmedel för att förmedla hur ett virus introduceras och sprids inom en population. Viktigt att notera är att detta arbete på grund av tidsrestriktioner bara hann utveckla grunderna till en sådan mjukvara. Projektet agerar som en grund varpå vidareutveckling av liknande mjukvara kan leda till ett inlärningsverktyg vilket kan hjälpa öka förståelsen för hur man kan agera under en framtida epidemi.

Det finns dock vissa saker som hade utvecklats annorlunda vid en nystart av ett liknande projekt. En av dessa är rörelsesystemet för icke-spelbara figurer. Som nämnt i avsnitt 3.2.3 använder figurerna ett navigationsnät för att röra sig. Implementationen av paketet ledde till problem vid vidare utveckling av funktioner som nyttjar rörelsesystemet. Exempelvis försvårade det implementation av social distansering mellan icke-spelbara figurer. Därför rekommenderar vi vid genomförande av liknande projekt att ett eget rörelsesystem utvecklas för att undvika komplikationer vid uppskalning av programmet.

Det rekommenderas även att en mer skalbar version av SIR-modellen används. I fall att man utvecklar en simulation för att observera ett virus på längre sikt bör man använda en SIR-modell där nya individer kan introduceras till populationen men även tas bort. Detta leder till man lättare kan skala programmet och implementera demografiska faktorer som ålder, kön, födsel, dödlighet och migration. En simulation med demografiska variabler kan få ett mer verklighetsbaserat resultat genom att exempelvis öka icke-spelbara figurers risk att bli smittade då de är äldre.

# Källor:


[1] W. O. Kermack och A. G. McKendrick, “A contribution to the mathematical theory of epidemics”, 1 Augusti 1927, Tillgänglig: https://royalsocietypublishing.org/doi/10.1098/rspa.1927.0118, Hämtad: 2021-05-24.

[2] H. S. Rodrigues, “Application of SIR epidemiological model: new trends”, 8 November 2016, Tillgänglig: https://arxiv.org/abs/1611.02565, Hämtad: 2021-05-24.

[3] E. Ortiz-Ospina, “How do people across the world spend their time and what does this tell us about living conditions”, December 08 2020, Tillgänglig: https://ourworldindata.org/time-use-living-conditions, Hämtad: 2021-05-24.

[4] Population Reference Bureau, “Average Household Size”, 2020, Tillänglig: https://www.prb.org/international/indicator/hh-size-av/map/region, Hämtad: 2021-05-24.

[5] Unity Technologies, “Unity Real-Time Development Platform | 3D, 2D, VR & AR ENGINE”, 2021, Tillgänglig: https://unity.com/, Hämtad: 2021-05-24.

[6] Epic Games, “The most powerful real-time 3D creation platform - Unreal Engine”, 2021, Tillgänglig: https://www.unrealengine.com/en-US/, Hämtad: 2021-05-24.

[7] Microsoft, “Visual Studio 2019 IDE - Programming Software for Windows”, 2021, Tillgänglig: https://visualstudio.microsoft.com/vs/, Hämtad: 2021-05-24.

[8] Github Inc., “Github”, 2021, Tillgänglig: https://github.com/, Hämtad: 2021-05-24.

[9] Github Inc,. “Github Desktop | Simple collaboration from your Desktop”, 2021, Tillgänglig: https://desktop.github.com/, Hämtad: 2021-05-24.

[10] Unity Technologies, “Unity - Scripting API: Canvas”, 15 Maj 2021, Tillgänglig: [https://docs.unity3d.com/ScriptReference/Canvas.html, Hämtad: 2021-05-24.

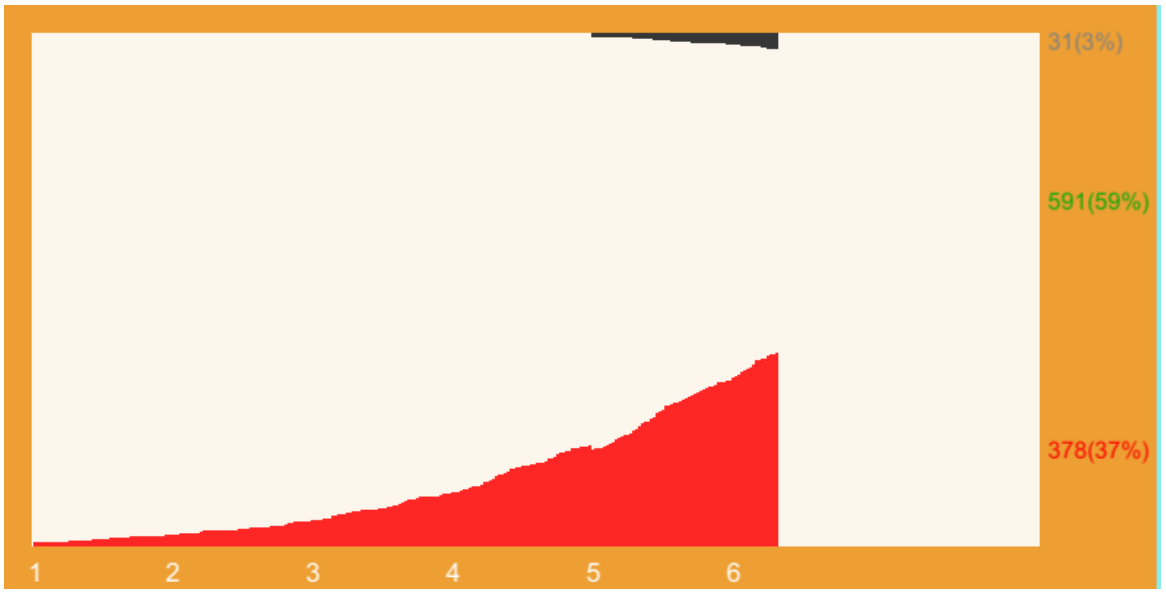
[11] Unity Technologies, “Unity - Scripting API: Vector2”, 15 Maj 2021, Tillgänglig: https://docs.unity3d.com/ScriptReference/Vector2.html, Hämtad: 2021-05-24.

[12] Unity Technologies, “Unity - Scripting API: RectTransform”, 15 Maj 2021, Tillgänglig: https://docs.unity3d.com/ScriptReference/RectTransform.html, Hämtad: 2021-05-24.

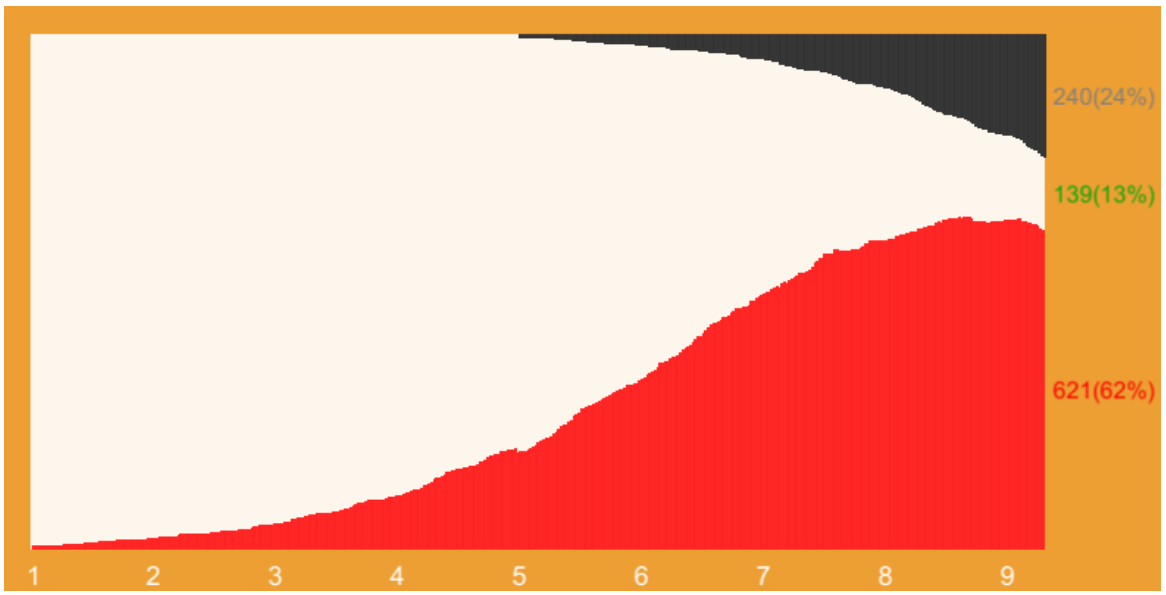
[13] J. Leng and D. R. Goldstein, “Impact of aging on viral infections”, 16 September 2010, Tillgänglig: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2998572/, Hämtad: 2021-05-24.

# Appendix:

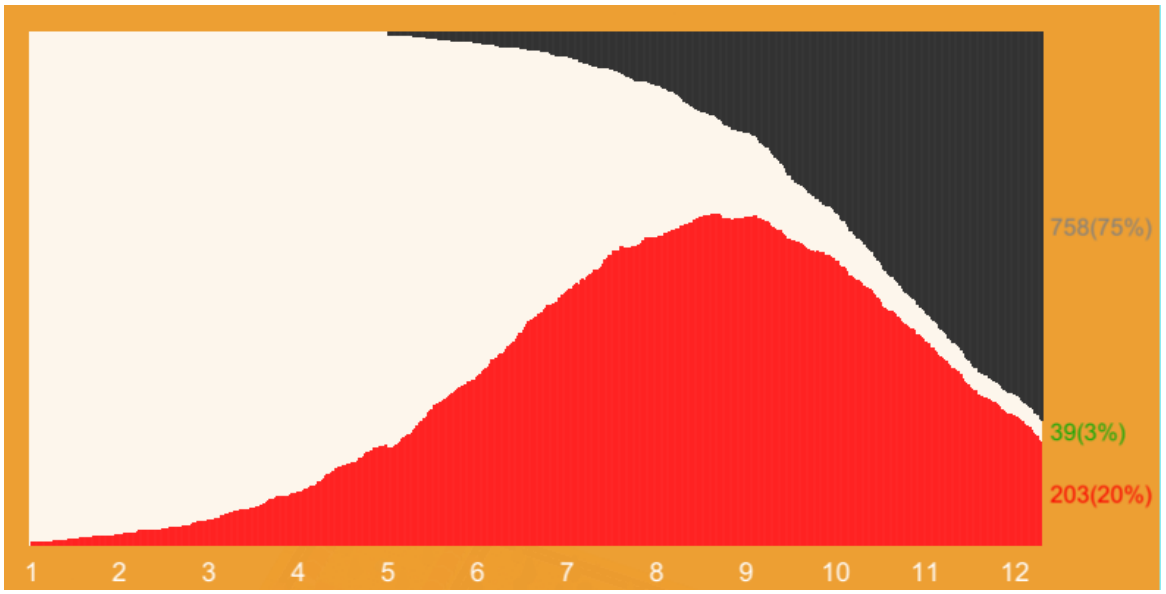
*Stapeldiagram över en simulering drygt tre dagar in. Notera att i ögonblicket bilden är tagen existerar det inga återhämtade icke-spelbara figurer.*



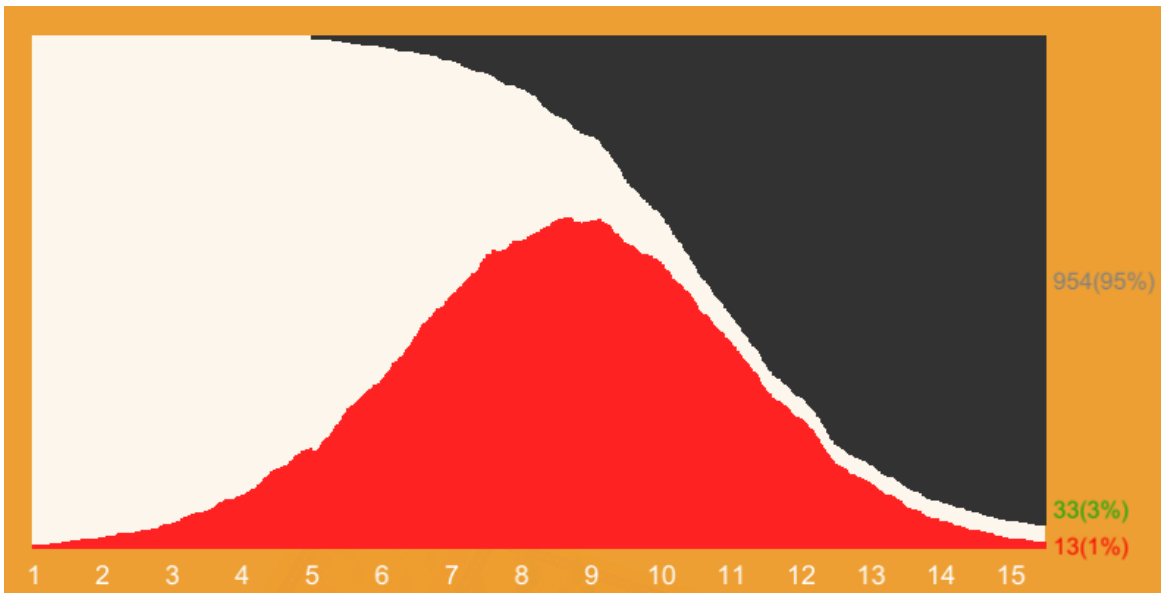

*Stapeldiagram över samma simulering drygt sex dagar in. De första icke-spelbara figurerna har nu återhämtat sig..*



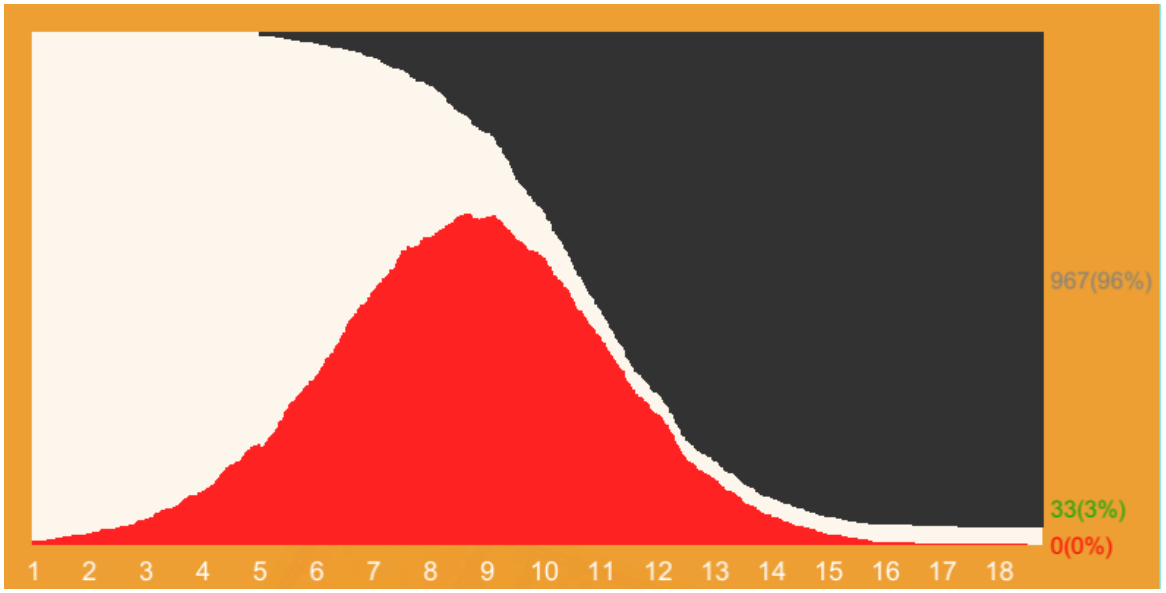

*Stapeldiagram över samma simulering drygt nio dagar in. Från diagrammet kan man anta att antalet infekterade nått sin topp.*




*Stapeldiagram över samma simulering drygt tolv dagar in. Antalet som återhämtar sig växer snabbare än antalet infekterade.*




*Stapeldiagram över samma simulering femton dagar in. Simuleringen börjar nå sitt slut.*




*Stapeldiagram över samma simulering arton dagar in. Viruset är utplånat och simulationen är färdig.*