A comprehensible interactive simulation of a pandemic in Germany

Advanced Pratical

30th May 2022

Marvin Hanf

Outline

1. Introduction

Covid-19 – The disease

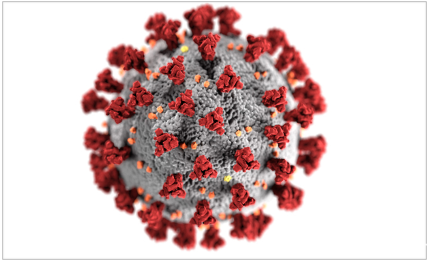


Figure 1.1

- Triggered by the pathogen SARS-CoV-2
- Contracts over droplets/aerosols
- Symptoms include coughing, fever and loss of taste/smell
- Causes respiratory infections, kidney infections, liver malfunctions and death by organ failure among other things
- Infections possible before symptoms begin to show, highest risk of infection shortly before and after breakout of symptoms

The Covid-19 Pandemic

- Disease first identified in Wuhan, China, roughly around the New Year of 2020
- Quickly spread around the world
- Worldwide over 500 million confirmed cases, 6.2 million deaths
- Over 11 billion vaccine doses administered

→ Response needed to keep death-rate and infect-rate at a minimum

Response Management

Measures have been taken against the spread of Covid-19:

- Masks in public spaces, filtering droplets and aerosols
- Lockdowns, e.g. closing restaurants and bars or even factories
- Curfews (mostly at night to counteract private parties)
- Meeting rules for every person
- Vaccinations to build up immunity and lower the spread

Simulating a Pandemic

Online-Simulation developed at the University of Saarland, Prof. Dr. Lehr et al. 2022

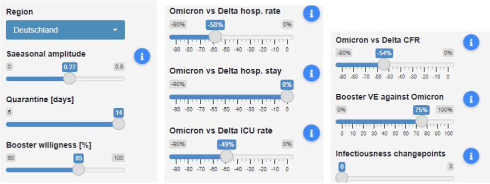


Figure 1.2: Settings

Figure 1.3: Results

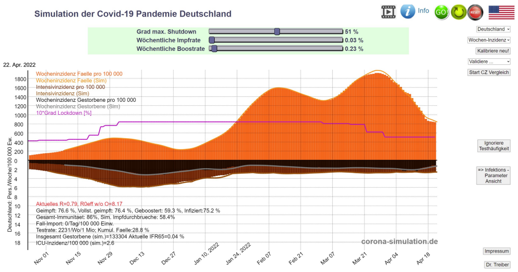


Figure 1.4: Online-Simulation by Dr. Treiber 2022

Figure 1.5: Game Plague Inc.

My motivation

- Simulators available are informative and close to reality
- Handling and interpretation difficult
- Not possible to change measures taken against the spread

→ Wanted to make simulation/serious game that is easier comprehensible and interactive while simulating
→ Responses to decrease the spread of Covid should be felt, to also understand the difficulty choosing the right measures

2. Models/Algorithm

2. Models/Algorithm

2.1 SIRD-Model

SIRD-Model

- S: Susceptibles, people that have not contracted the virus
- I: Infected, people that have contracted the virus
- R: Recovered, people that have recovered from the virus
- D: Dead, people that have died because of the virus

→ Enables the modelling of a simple simulation

SIRD-Model

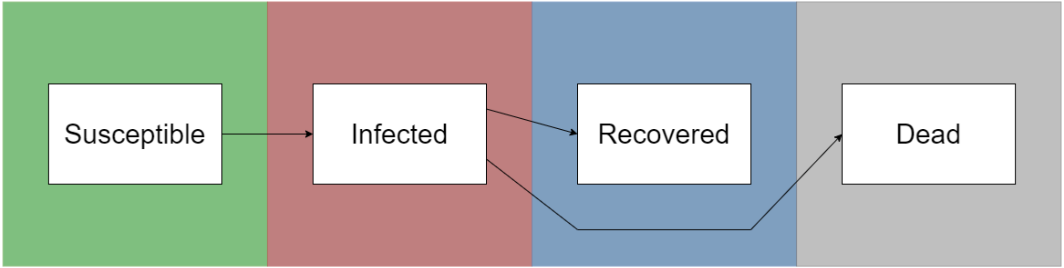


Figure 2.1: Flowchart of the SIRD-Model

2. Models/Algorithm

2.2 Gillespie-Algorithm

Stochastic Algorithms

- Fitting algorithm is needed
- Data has to be statistically coherent

Considered two different approaches:

→ **ODEs:** A system of ordinary differential equations

→ **Gillespie-Algorithm:** A system of reaction rates and rules

Stochastic Algorithms (Dis-)Advantages

ODEs:

+ ODEs can be solved quickly by a computer
- Some rounding issues may occur

Gillespie-Algorithm:

+ Results calculated to the full integer
+ Easily scalable, when introducing new rules
- Complicated models need much computing capacity

Biggest difference: ODEs just calculate the data results, whereas the Gillespie-Algorithm simulates every event (e.g. an infection) explicitly

Gillespie-Algorithm

- Simulates a model in a predefined time interval, e.g. one day, or until a certain state inside the model is reached
- Based on a predefined set of reaction rates and rules
- Rules will carry out the simulated event
- Two random floats needed:
    1. float continues the time
    2. float chooses the rule to execute
- New rates & rules can be added easily

Gillespie-Algorithm Steps

1. Generate two random floats $r_1, r_2 \in [0, 1]$
2. Calculate reaction rates, the total sum and the cumulative sum of the reaction rates
3. Advance the time $t = t + \Delta t$, Δt is in dependence of $\frac{r_1}{\text{Total Rate Sum}}$, r_1 can be modified by a function, e.g. $-log(r_1)$, so that the number of events better fit the desired time frame.
4. Divide every value of the cumulative sum by the total sum to place it on the interval $[0, 1]$
5. Determine the rule that will be executed with the second float by placing it into the cumulative sum interval
6. Execute the rule
7. Repeat until time limit is reached or desired model state is reached

Gillespie-Rules Example with SIRD

Rules and Rates:

$I:$ A susceptible gets infected: $S + I \xrightarrow{\iota} 2I$ with rate $\frac{\iota}{N} SI$

$R:$ An infected recovers: $I \xrightarrow{\rho} R$ with rate ρI

$D:$ An infected dies: $I \xrightarrow{\delta} D$ with rate δI

ι is the infect rate of the disease, ρ the recovery rate, δ the death rate and N the sum of people

2. Models/Algorithm

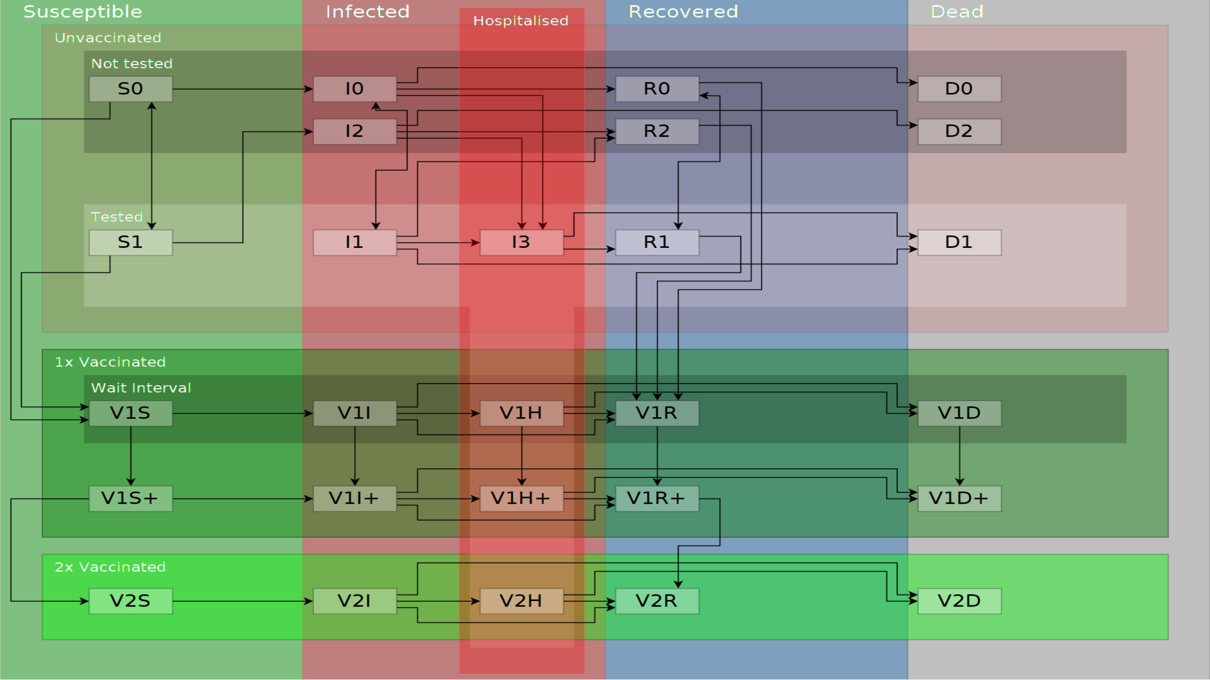
2.3 My extended SIRD-Model

Gillespie-Rules for the extended SIRD

Rules and Rates (excerpt):

- I : A susceptible gets infected: $S + I \xrightarrow{\iota} 2I$ with rate $\frac{\iota}{N}SI$
- R : An infected recovers: $I \xrightarrow{\rho} R$ with rate ρI
- D : An infected dies: $I \xrightarrow{\delta} D$ with rate δI
- H : An infected gets hosptialised: $I_0 \xrightarrow{\eta} I_3$ with rate $\eta\frac{B_{max} - I_3}{N} I_0$
- V : A susceptible gets vaccinated: $V + S \xrightarrow{\omega} S_V$ with rate ωVS
- T : An unvaccinated susceptible gets tested $S_0 \xrightarrow{\vartheta} S_1$ with rate ϑS_0.
  A tested susceptible loses the test status $S_1 \xrightarrow{\lambda} S_0$ with rate λS_1

ι: infect rate, N: total number of people, ρ: recovery rate, δ: death rate, η: hospitalisation rate, ω: vaccination rate, V: available vaccination doses, ϑ: testing rate, λ: loss of testing rate



Susceptible
Infected
Hospitalised
Recovered
Dead
Unvaccinated
Not tested
S0
I0
R0
D0
I2
R2
D2
Tested
S1
I1
I3
R1
D1
1x Vaccinated
Wait Interval
V1S
V1I
V1H
V1R
V1D
V1S+
V1I+
V1H+
V1R+
V1D+
2x Vaccinated
V2S
V2I
V2H
V2R
V2D

Extended SIRD-Model

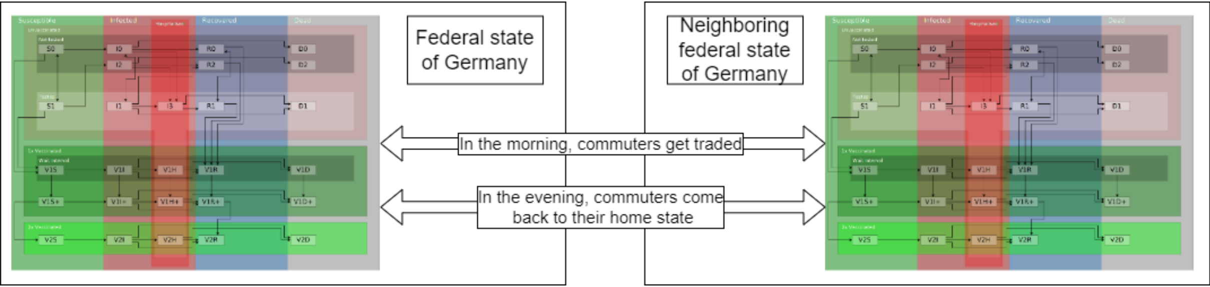


Figure 2.3: Random commuters get sorted into their respective category in the visiting state and act accordingly

Assumptions made

- Once recovered, no reinfection possible
- Every individual is willing to be vaccinated
- Individuals will not be vaccinated while infected
- Individuals in the hospital are always tested
- Positive tested individuals are less likely to infect others than non-tested infected
- Individuals with unknown status change aren't tested again
- Commuters commute before the first Gillespie-event takes place and go back home after the last
- Positive tested individuals don't commute, 1x Vaccinated individuals inside the wait period also don't commute
- Commuters are not able to be hospitalized
- Vaccination of commuters only possible in their own federal state
- Tests are able to determine if a person was already infected
- For each death it gets determined if the cause was the disease or not

3. Code

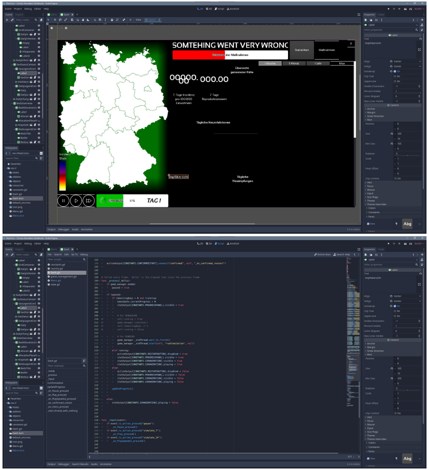


Figure 3.1: Editor

- Open-Source Game Engine
- GDScript, Python-like scripting language
- Nodes for various uses
- Flexible Scene-System, Nodetrees
- Visual Editor for Scenes
- Easy deployment to many platforms
- Comes with a multi-platform editor
- Much more features

3. Code

3.1 Structure

Basic Structure

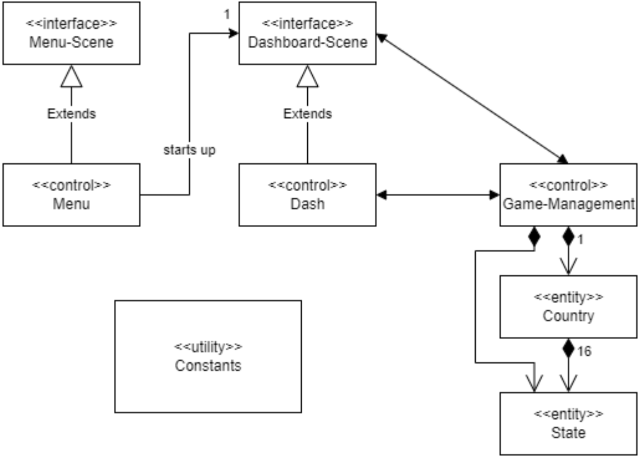


Figure 3.2: Overview over the basic structure

- Two Scenes
- Five Base Classes
- One static class for Constant Expressions and some static functions

Basic Structure

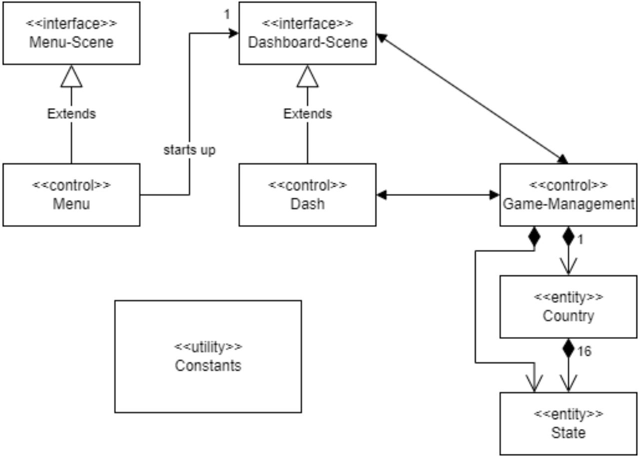


Figure 3.3: Overview over the basic structure

- Classes that extend the scenes contain the main threads
- Game-Management handles Input from Dashboard-Scene and generates the output
- Country and State model the real-life, simulate and store data
- State contains main simulation function

3. Code

3.2 Code of the Simulation

Main-Thread inside Dash Control Class

```
func _process(_delta): # called every frame, _delta is the elapsed time between two frames
        if !paused:
                if remainingDays > 0 and !running:
                        Constants.currentProgress = 0
                        statOutput[CONSTANTS.PROGRESSPANEL].visible = true
                        # For DEBUGGING
#                       self.running = true
#                       game_manager.simulate()
#                       self.remainingDays -= 1
#                       self.running = false
                        # For RUNNING
                        game_manager._simThread.wait_to_finish()
                        game_manager._simThread.start(self, "runSimulation", null)

                updateProgress()
```

Inside Game-Management Class

```
func simulate():
        entities[CONSTANTS.DEU].simulateALL()
        if self.days.size() > 10:
                var checkNewInfections = []
                for i in range(self.days.max() - int(CONSTANTS.MONTH * CONSTANTS.
                     ENDEMICTIMEFACTOR), self.days.max()):
                        if i < 2:
                                continue
                        else:
                                checkNewInfections.append(entities[CONSTANTS.DEU].
                                     getDailyInfections(i, true))
                if checkNewInfections.max() < 1:
                        setMode(CONSTANTS.ENDMODE)
                        if !ended:
                                endDay = currentDay
                        print("Day ", self.currentDay, ": PANDEMIC OVER")
        updateDay()
```

Inside the Country Class

```
func simulateALL():
	produceVax()
	distributeVax()
	distributeCommuters()

	for state in states.values():
#		state.simulate() # for easier debugging
		state._thread.start(self, "simulateState", state.getName())
	for state in states.values():
		state._thread.wait_to_finish()

	homeCommuters()
	for state in states.values():
		state.collectNumbers()
	getNumbers()
	recalculateStatePopulation()
```

Distributing Commuters

- Commuter-Rates based on data from the ’Agentur für Arbeit’
- Commuters get evenly distributed to neighboring states in round-robin style
- Randomly selected one by one from a category
- Positive tested, hospitalised and individuals inside the waiting interval don’t commute

Inside the State class

```
func simulate():
        var startTime = OS.get_ticks_msec()
        events = 0
        infectRate = [getInfectRate(), getInfectRate()*infectTestFactor, getInfectRate()*
             infectFactorHosp, getInfectRate()*infectFactorV1, getInfectRate()*infectFactorV2]
             # untested, tested, hospitalised, 1x-vaxed, 2x-vaxed
        var t = timeDifference
        while t<1:
                t = gillespieIteration(t)
                events += 1
                if(t>1):
                        timeDifference = fmod(t,1)
                        continue
        waitDay += 1
        waitDay = waitDay % Constants.VACDELAY
        V1eligible[0] += V1[0][waitDay]
        V1[0][waitDay] = 0
        V1eligible[1] += V1[1][waitDay]
        V1[1][waitDay] = 0
        # and so on
```

Gillespie-Iteration

```
func gillespieIteration(t):
        var r1 = rnd.randf()
        var reactionRates = updateReactionRates()
        var reactTotal = CONSTANTS.sum(reactionRates)
        if reactTotal == 0:
                return 1
        var waitTime = -log(r1)/reactTotal
        t = t + waitTime
        var r2 = rnd.randf()
        var reactionRatesCumSum = CONSTANTS.cumulative_sum(reactionRates)
        for i in range(reactionRatesCumSum.size()):
                reactionRatesCumSum[i] = reactionRatesCumSum[i] / reactTotal
        var rule
        for i in range(reactionRatesCumSum.size()):
                if(r2 <= reactionRatesCumSum[i]):
                        rule = i
                        break
        updatePersonNumbers(rule)
        return t
```

Reaction-Rates [1]

```
func updateReactionRates():
  var rates = []

  # 0 1 2 3 Infection of untested, non-vaxed individuals
  rates.append((infectRate[0]/population)*S[0]*I[0])  # by untested infected (non-vaxed)
  rates.append((infectRate[1]/population)*S[0]*I[1])  # by tested infected (non-vaxed)
  rates.append((infectRate[0]/population)*S[0]*I[2])  # by unknowingly infected (non-vaxed)
  rates.append((infectRate[2]/population)*S[0]*I[3])  # by non-vaxed hospitalised

  # many more rules, 149 in total
  return rates
```

[1] Bauer et al. 2021

Updating the individual numbers

```
func updatePersonNumbers(rule):
  match rule:
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12: # Infection untested non-vaxed
      S[0] -= 1
      I[0] += 1
    13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25: # Infection tested non-vaxed
      S[1] -= 1
      I[2] += 1
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38: # Infection 1x-vaxed, that are in the waiting interval
      var randomDay = rnd.randi() % Constants.VACDELAY # assign infection randomly to a waiting-block
      while true:
        if V1[0][randomDay] > 0:
          break
        else:
          randomDay += 1
          randomDay = randomDay % Constants.VACDELAY
      V1[0][randomDay] -= 1
      V1[1][randomDay] += 1
    # 149 rates have to be matched to rules
```

Saving the data

```
var sus0 = [CONSTANTS.NTESTED + CONSTANTS.SUSCEPTIBLE]  # untested susceptible
var sus1 = [CONSTANTS.TESTED + CONSTANTS.SUSCEPTIBLE]   # tested susceptible
var inf0 = [CONSTANTS.NTESTED + CONSTANTS.INFECTED]     # untested infected
var inf1 = [CONSTANTS.TESTED + CONSTANTS.INFECTED]      # tested infected
var inf2 = [CONSTANTS.UNAWARE + CONSTANTS.INFECTED]     # unaware infected
var hosp = [CONSTANTS.HOSPITALISED]                     # non-vaxed hosptialised
var rec0 = [CONSTANTS.NTESTED + CONSTANTS.RECOVERED]    # untested recovered
var rec1 = [CONSTANTS.TESTED + CONSTANTS.RECOVERED]     # tested recovered
var rec2 = [CONSTANTS.UNAWARE + CONSTANTS.RECOVERED]    # unaware recovered
var dead0 = [CONSTANTS.NTESTED + CONSTANTS.DEAD]        # untested dead
var dead1 = [CONSTANTS.TESTED + CONSTANTS.DEAD]         # tested dead
var dead2 = [CONSTANTS.UNAWARE + CONSTANTS.DEAD]        # unaware dead
var vax1sus = [CONSTANTS.VAX1 + CONSTANTS.SUSCEPTIBLE]  # 1x-vaxed susceptible
var vax1inf = [CONSTANTS.VAX1 + CONSTANTS.INFECTED]     # 1x-vaxed infected
var vax1hosp = [CONSTANTS.VAX1 + CONSTANTS.HOSPITALISED]# 1x-vaxed hosptialised
var vax1rec = [CONSTANTS.VAX1 + CONSTANTS.RECOVERED]    # 1x-vaxed recovered
var vax1dead = [CONSTANTS.VAX1 + CONSTANTS.DEAD]        # 1x-vaxed dead
var vax2sus = [CONSTANTS.VAX2 + CONSTANTS.SUSCEPTIBLE]  # 2x-vaxed susceptible
var vax2inf = [CONSTANTS.VAX2 + CONSTANTS.INFECTED]     # 2x-vaxed infected
var vax2hosp = [CONSTANTS.VAX2 + CONSTANTS.HOSPITALISED]# 2x-vaxed hospitalised
var vax2rec = [CONSTANTS.VAX2 + CONSTANTS.RECOVERED]    # 2x-vaxed recovered
var vax2dead = [CONSTANTS.VAX2 + CONSTANTS.DEAD]        # 2x-vaxed dead
```

4. Introduction/Demonstration of the software

Simulating – Options

- Simulating 7 or 14 days at once
- Pause Simulation
- Toggling Godmode (toggling between full data and measured data through tests)
- Choosing a simulation-factor, that determines how much computing power is needed and correlates with the quality of the simulated data
- Restart Simulation from zero with same random seed

Available Measures against the spread of the disease

Measures available for each federal state:

- Lockdowns with various strictness that include:
  - Four Masking options
  - Four Remote-Working options
- Closing the borders of a federal state
- Four testing options for unvaccinated people

$\rightarrow$ Measures are written in figurative speech, e.g. 'Workers are obligated to work from home' or 'Unvaccinated people need a test to enter public buildings'
$\rightarrow$ State-measures can be overwritten by the country

Additional country measures:

- Adjust the number of available intensive care beds
- Adjust the daily production of vaccines

Available Measures against the spread of the disease

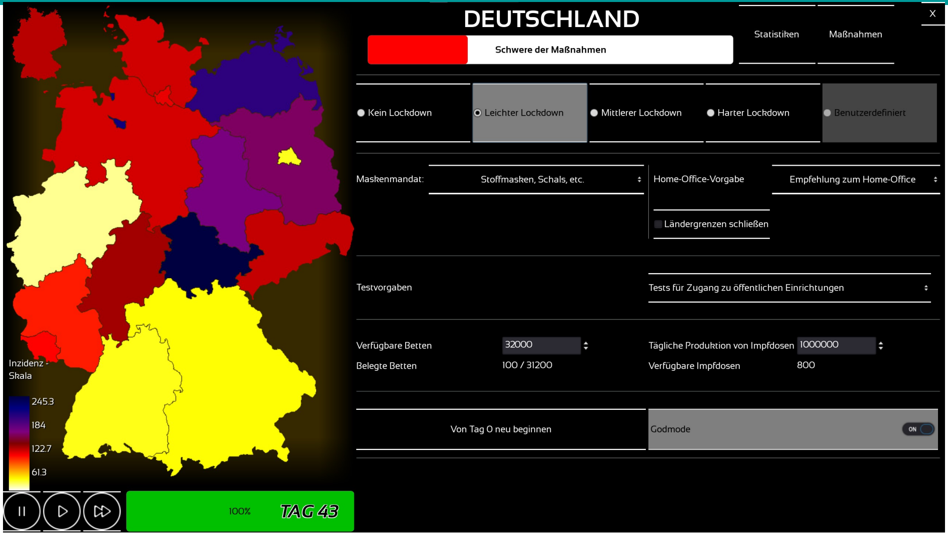


Figure 4.1: Action Menu

Implementing the measures

```
func simulate():
        var startTime = OS.get_ticks_msec()

        events = 0

        infectRate = [ getInfectRate() , getInfectRate() *infectTestFactor, getInfectRate() *infectFactorHosp, getInfectRate() *infectFactorV1, getInfectRate() *infectFactorV2] # untested, tested, hospitalised, 1x-vaxed, 2x-vaxed
        var t = timeDifference

        while t<1:
                t = gillespieIteration(t)
                events += 1
                if(t>1):
                        timeDifference = fmod(t,1)
                        continue


        waitDay += 1
        waitDay = waitDay % Constants.VACDELAY
```

Infect-Rate and Commuter-Rate [2]

```
func getInfectRate():
	if self.selectedMask != 0 or self.selectedHomeOffice != 0:
		var lockdownStrictness = (CONSTANTS.LOCKDOWNSTRICTNESS[self.selectedHomeOffice] + (2 * CONSTANTS.MASKFACTORS[self.selectedMask])) / 3.0
#		print(lockdownStrictness)
		return baseInfect * (1-lockdownStrictness)
	else:
		return baseInfect

func getCommuteRate():
	return self.commuterRate * CONSTANTS.COMMUTERFACTORS[self.selectedHomeOffice]
```

[2] **manica2021effectiveness**, Bagheri et al. 2021, DESTATIS 2022a

Available Stats

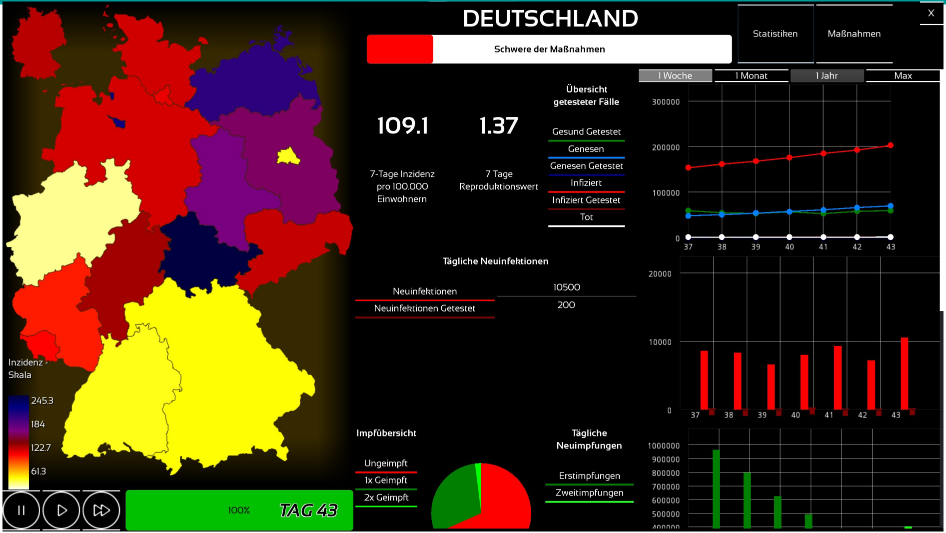
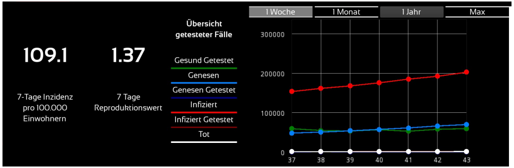
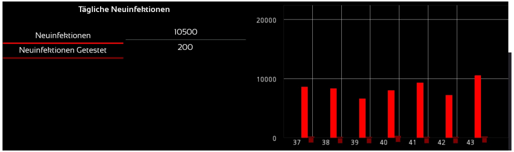


Figure 4.2: Stats Overview

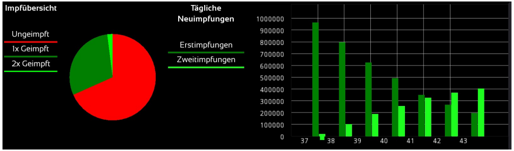
Available Stats



(a) Overview over all events

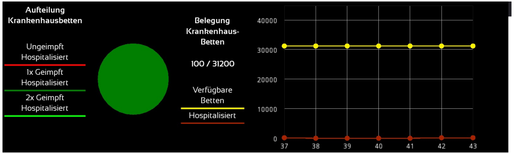


(b) Daily new infections

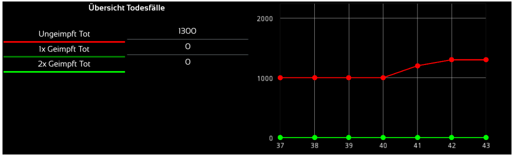


(c) Overview over vaccination progress and daily new vaccinations

Figure 4.3



(a) Overview over hospitalisations and the vaccine status of the hospitalised



(b) Overview over deaths and vaccine status of the deceased

Figure 4.4

Overall-Indicator behind map

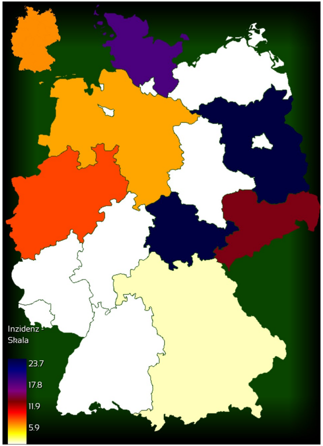


Figure 4.5: Green Indicator

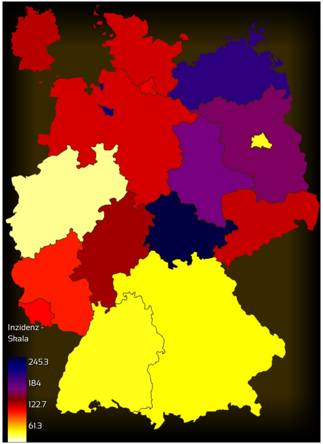


Figure 4.6: Yellow Indicator

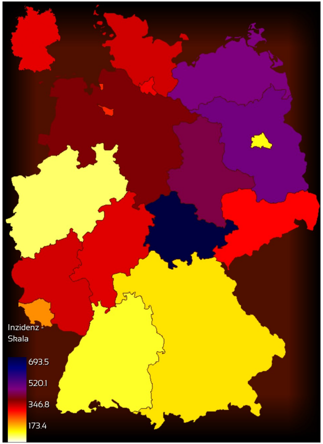


Figure 4.7: Red Indicator

End of the simulation – Available stats

- Number of days of how long your pandemic has lasted
- How many have been infected, recovered, died
- Number of administered vaccines and how many people have been vaccinated
- Number of deaths and their vaccination status
- Day of most active cases and day with the most new infections
- Overall Overview of the cases

→ Quick Restart with same random seeds possible

End of the simulation

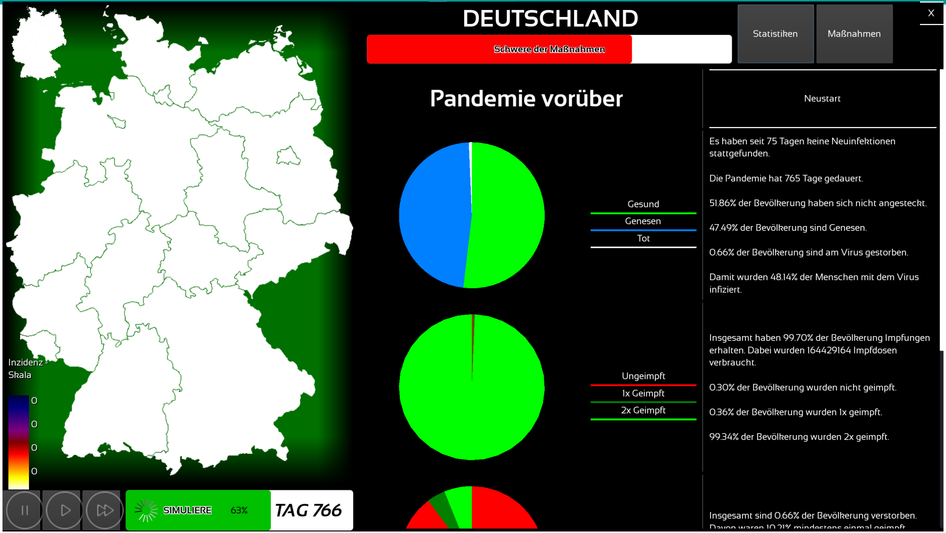


Figure 4.8: Endscreen with statistics

Demonstration

5. Future Ideas/Conclusion

Future Ideas

- Money as a gameplay feature (Tests and vaccination cost money, lockdowns reduce income)
- Age groups
- Vaccination Research as gameplay element

Conclusion

- Statistically coherent simulation
- User can easily interact with the simulation
- Interactivity gives user a feeling on how the dynamics of a pandemic work, and how hard it is to choose the right counter-measures
- Simulation adjustable to the computing power of user's machine, pandemic development will always have the same dynamic, but the given numbers may differ

Download

Software available for Windows, MacOS and Linux.

Downloadlink:

`https://heibox.uni-heidelberg.de/d/6c36d27cc85244f1870e/`
and soon available at `https://pille.iwr.uni-heidelberg.de/`

Contact: marvin.hanf@stud.uni-heidelberg.de

Feedback

Thank you for listening!

Questions?

Image Sources

Figure 1.1 `https://www.pei.de/DE/newsroom/dossier/coronavirus/coronavirus-node.html`

Figure 1.2 Prof. Dr. Lehr et al. 2022

Figure 1.3 Prof. Dr. Lehr et al. 2022

Figure 1.4 Dr. Treiber 2022

Figure 1.5 `https://store.steampowered.com/app/246620/Plague_Inc_Evolved/?l=german`

Images not listed here were taken or created by myself.

References

Prof. Dr. Lehr, Thorsten et al. (2022). *CoSim, Online COVID-19 Simulator*. URL: https://covid-simulator.com/ (visited on 04/21/2022).

Dr. Treiber, Martin (2022). *Simulation der Covid-19 Pandemie in Deutschland*. URL: https://corona-simulation.de/ (visited on 04/21/2022).

Bauer, Steffen et al. (2021). *A rule-based epidemiological framework for modelling and simulation in the context of the covid-19 pandemic*. DOI: 10.48550/ARXIV.2111.07336. URL: https://arxiv.org/abs/2111.07336.

Bagheri, Gholamhossein et al. (2021). “An upper bound on one-to-one exposure to infectious human respiratory particles”. In: *Proceedings of the National Academy of Sciences* 118.49, e2110117118. DOI: 10.1073/pnas.2110117118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2110117118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2110117118.

DESTATIS (2022a). *Mobilitätsindikatoren auf Basis von Mobilfunkdaten*. URL: https://www.destatis.de/DE/Service/EXDAT/Datensaetze/mobilitaetsindikatoren-mobilfunkdaten.html#Pendlerverhalten (visited on 05/23/2022).

Manica, Mattia et al. (2021). “Impact of tiered restrictions on human activities and the epidemiology of the second wave of COVID-19 in Italy”. In: *medRxiv*. DOI: 10.1101/2021.01.10.21249532. URL: https://www.medrxiv.org/content/early/2021/02/24/2021.01.10.21249532.

DESTATIS (2022b). *Presse: Zahl der Intensivbetten im Jahresdurchschnitt 2020 um 5 % höher als im Vorjahr*. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2021/12/PD21_585_231.html (visited on 05/23/2022).

Bundesagentur für Arbeit, BA (2022). *Pendlerverflechtungen der sozialversicherungspflichtig Beschäftigten nach Ländern - Deutschland (2020)*. URL: https://statistik.arbeitsagentur.de/SiteGlobals/Forms/Suche/Einzelhef