No Planet B

A Simulation Game

Johannes Bohlig and Dominik Buchmann

Outline

- Motivation and Game Idea
- Planet Generation
- Planet Gamification
    - Continent Creation
    - Classification
    - Planet - User Interaction
    - Moveables
- Simulation
    - Main Simulation Loop
    - Planet Handling
    - Interface and Events
    - Spawnables
- Demonstration
- Discussion

Motivation & Game Idea

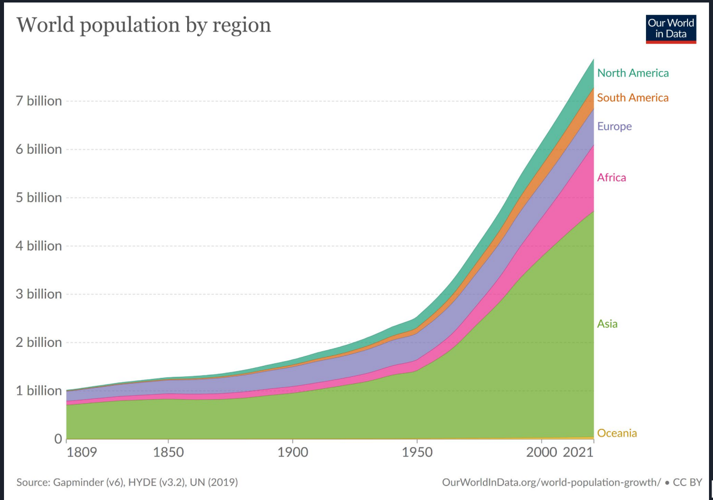
Motivation

- Recent Topic in Media and Society
- What are the reasons ?
- What aspects do we simulate ?

[1]

[2]

Motivation

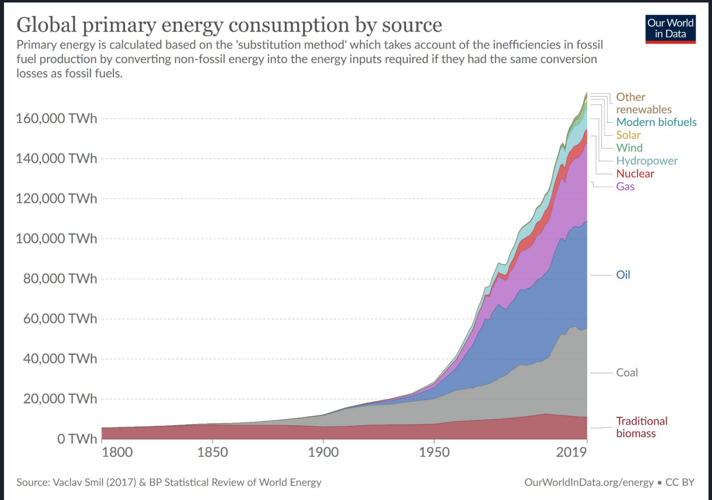


Source: Gapminder (v6), HYDE (v3.2), UN (2019)

OurWorldInData.org/world-population-growth/ • CC BY

[3]

- Rising Population



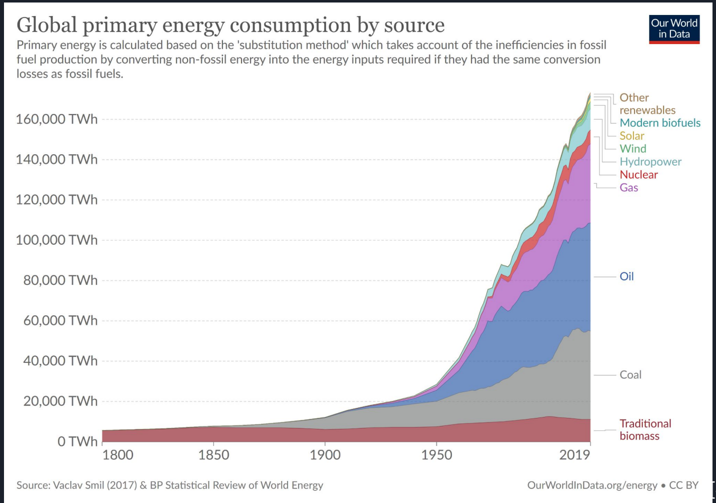
Source: Vaclav Smil (2017) & BP Statistical Review of World Energy

OurWorldInData.org/energy • CC BY

[4]

- Rising Energy Demand

Motivation

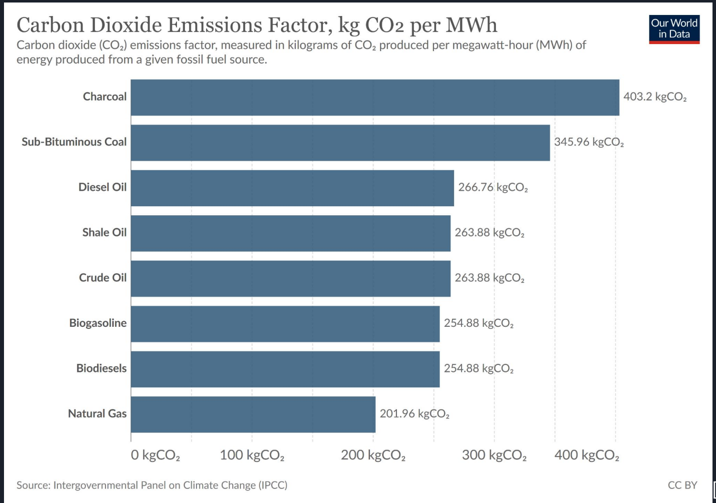


Global primary energy consumption by source

Primary energy is calculated based on the 'substitution method' which takes account of the inefficiencies in fossil fuel production by converting non-fossil energy into the energy inputs required if they had the same conversion losses as fossil fuels.

Source: Vaclav Smil (2017) & BP Statistical Review of World Energy

OurWorldInData.org/energy • CC BY [4]



Carbon Dioxide Emissions Factor, kg CO2 per MWh

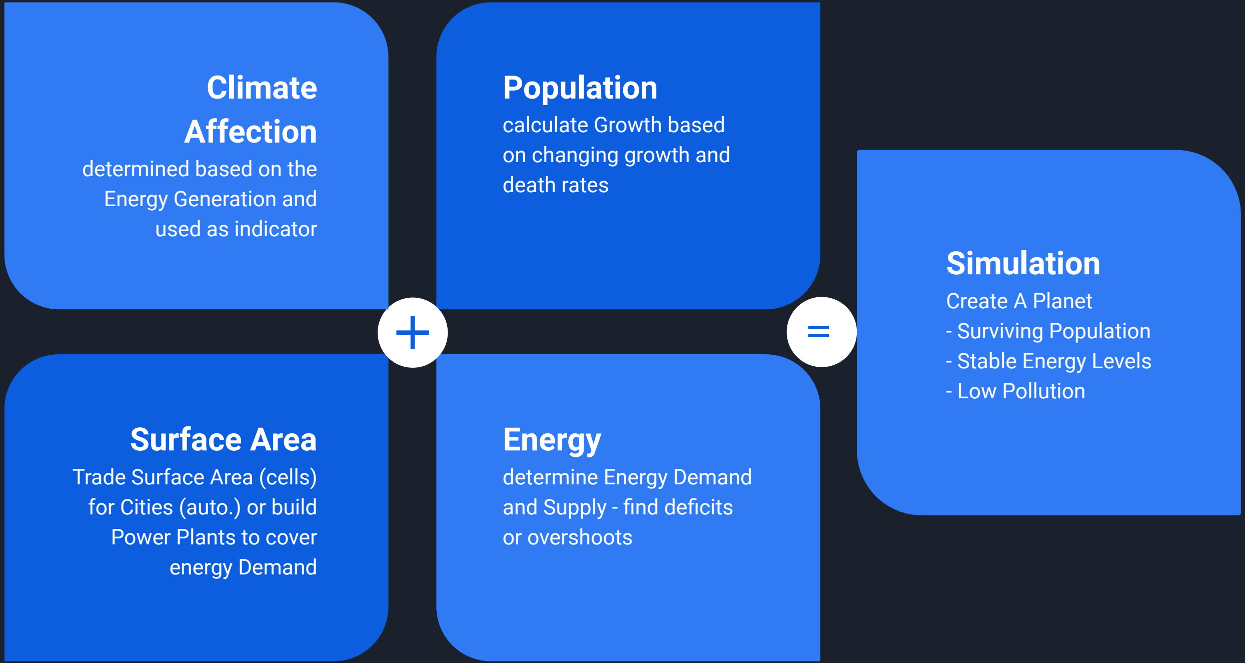
Carbon dioxide (CO_2) emissions factor, measured in kilograms of CO_2 produced per megawatt-hour (MWh) of energy produced from a given fossil fuel source.

Source: Intergovernmental Panel on Climate Change (IPCC)

CC BY [5]

- Rapid growth in “cheap” energy sources that produce high CO^2 Levels

Game Idea



Climate Affection
determined based on the Energy Generation and used as indicator

Population
calculate Growth based on changing growth and death rates

+

Surface Area
Trade Surface Area (cells) for Cities (auto.) or build Power Plants to cover energy Demand

Energy
determine Energy Demand and Supply - find deficits or overshoots

=

Simulation
Create A Planet
- Surviving Population
- Stable Energy Levels
- Low Pollution

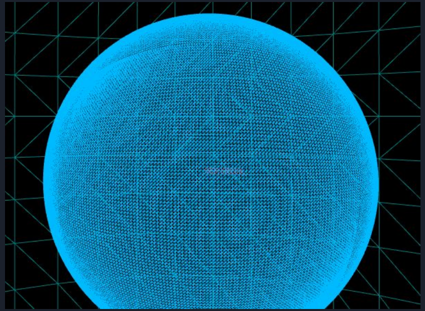
Planet Generation

Planet Generation

- Cube as basis
    - n * n * 6 points
        - n = edge vertices
        - 6 sides

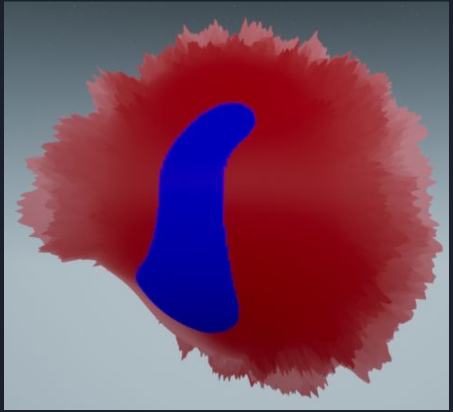
- Subdivide edges into mesh

- Normalize distance from origin of every point in mesh
    - result in sphere
    - More precise way is used by the formula [6]:
        - $x = x*sqrt(1-(y^2 + z^2)/2 + (y^2*z^2)/3)$
        - $y = y*sqrt(1-(z^2 +x^2)/2 + (z^2*x^2)/3)$
        - $z = z*sqrt(1-(x^2 + y^2)/2 + (x^2*y^2)/3)$



Planet Generation

- Manipulate every point location with a noise function to gain landmasses
  - 3D perlin noise
  - done twice on top of landmasses to gain mountain formations



- Triangulate the resulting mesh to render
  - use point locations also as normals
  - calculate UVs based on point location on sphere
    - map 3D - position to [longitude, latitude] on sphere

- Planet Generation based on the Unity-Series[7] of Sebastian Lague

Planet Gamification

- Generated Planet is basis for the playground

- To get a usable playground we need to:
  - Classify cells (Water, Land, Mountain, etc…)
  - Create continuous Surface
  - Identify Continents on the surface
  - Interact with the planet, continent and cells
  - Need to move objects seamless on the surface

Classification

- Classification by height measure
    - height = distance of point to origin
    - measure maximum height of all points
    - classification:
        - under ⅓ of max height is classified as water
        - ⅓ to ⅔ of max height is classified as land
        - over ⅔ of max height is classified as mountain

- Points of same class get leveled
    - water height 0
    - land height 0.5*maxHeight

Continuous Surface

- All Sphere Surface points are stored in 1D-Array
    - Those points are the surface cells
- To execute actions and move seamlessly around the cube/sphere
    - mapping from 1D array of points to cube
- ´BorderlessArrayAccess´ and ´BorderlessIndexMap´ - methods developed
    - contain a static mapping of all Cube sides to their neighbor sides
        - consists of:
            - offset to neighbor in 1D-Array
            - turning values in creation direction

Identification of Continents

- Visually disjoint surface areas need to be logically separated

- Every Continent should be stored as own object

- Therefore the “Continent Creation Algorithm” was developed

Continent Creation Algorithm

Multiple Stages:

- Stage 1:
  - Every cell of the cube is traversed by an kernel (3 x 3 matrix)
    - Every side is traversed line by line
  - The cells is numbered depending on it's classification:
    - -1 if it has an classification as water
    - A positive number if it's classified as land (land, mountain, etc...)
    - Special case:
      - if a cell in the kernel has already a positive number, the current cell is saved as a pair with the (lowest) number in the kernel -> belong to the same continent {1,2}
    - Every time a continent is left, the "polygon-counter" is increased
    - To cover the edge cases of every cube-side the "BorderlessArrayAccess" - Method is used

- Stage 1 results in many associated pairs of polygon pairs
- Since the cube sides are traversed from top to bottom different cases are missed where continents belong together

Continent Creation Algorithm

- Stage 2:
    - Associations between the found continent pieces have to be made
    - 3 different association cases:
        - Case 1:
            - Pairs that are found by the kernel in stage 1 and have the same first value
                - {1,2} ; {1,3} => {1,2,3}
        - Case 2:
            - Pairs or associations that have their first pair value in as a second (or higher) value
                - {1,2}; {2,3,4} => {1,2,3,4}
        - Case 3:
            - Pairs or associations that have a different first value but same second value
                - {1,4};{2,4} => {1,2}
    - Those have to be made separately in the given order 1,2,3 since one association can build up another
        - Have to be made, till nothing changes
    - Array iteration from back to front to find associations faster

Continent Creation Algorithm

- Stage 3:
    - Final associated continent values are assigned to the according cells
    - Continent values are used as continent ID
    - Continents are colored depending on their ID

User-Planet Interaction

- Every cell of the planet has to be interactable by the mouse

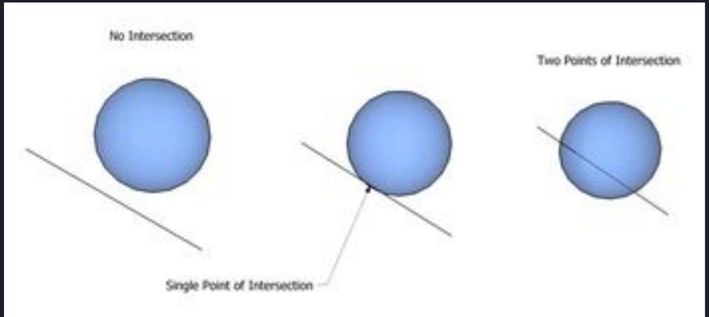
- Unreal Engine Collision Detection can’t be utilized due to performance and usability reasons
    - huge number of cells has to be addressed, up to many thousand

- “Ray-Sphere Intersection“ - Algorithm is used

Ray-Sphere Intersection

- Shoot Ray from Camera Position

- Determine Intersection Point by solving the equation system



[8]

- If there are one or two solutions, the ray has intersected with the sphere

User-Planet Interaction

- Ray has to be shot from the Users 2D-mouse Position
  - Deproject the 2D-mouse position at camera position into 3D world coordinates
- Use resulting Ray to Intersect with the sphere

- Problem: Different Cell heights cause precision problems
- Solution: Use multiple spheres with different radii to compare with
    - measure the distances of the resulting points of the ray sphere intersections with the original sphere point coordinates
    - compare distances and take the smallest as collision point

Moveables

- Need to move to more than one neighboring surface seamlessly

- Moveables based on “BorderlessArrayAccess”-Function
  - exploiting the function

- Own class to inherit from

Moveables

- Algorithm to track path of objects relative to their origin coordinates
  - objects can use their normal x and y coordinates to define position on planet surface

- Algorithm uses tracking coordinates internally
  - stored in the Moveable-Object

- relative coordinates “bent like a wire”
  - while object coordinates stay on a straight line

Moveables

- Make use of the existing static mapping of the cube sides
- When variable x or y of the object get increased a corresponding mapping coordinate will also be increased
- The current direction of the mapping coordinate is stored in the object
- When the mapping coordinate reaches the border of a cube side the mapping will be changed
    - the running direction of the mapping coordinates is changed based on mapping values of the static mapping

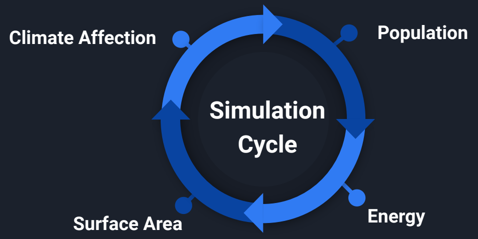
Moveables

- Mapping values are collected in a list

- All collected mapping values are applied to the object x and y coordinates
    - -> resulting in a line of coherent mapped cells

Simulation

Simulation

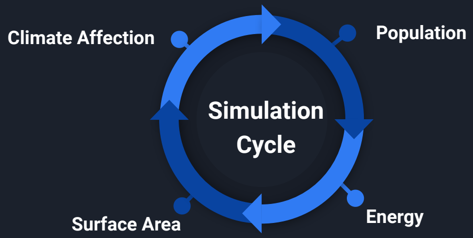


- Continuous Loop (Pausable)
    - Manage Planet State Structure

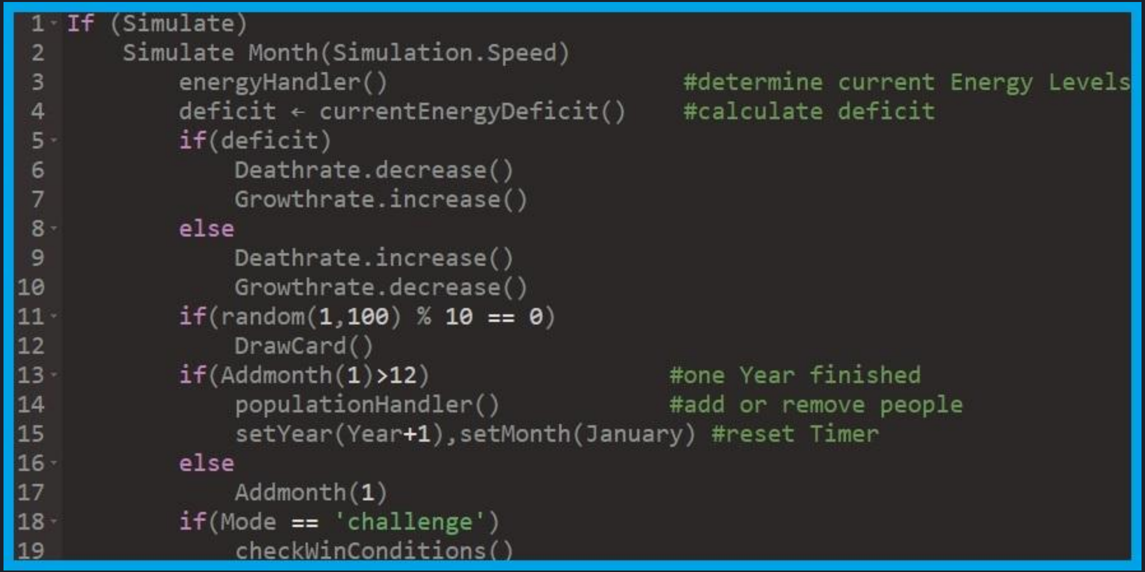
- Adjustable Speed (Timer independent of Framerate)

- Endless or Challenge Mode

Simulation



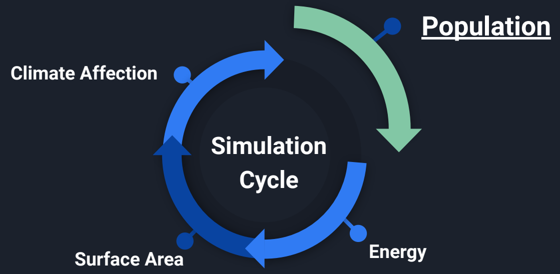
- Continuous Loop (Pauseable)



```
If (Simulate)
    Simulate Month(Simulation.Speed)
        energyHandler()                     #determine current Energy Levels
        deficit ← currentEnergyDeficit()    #calculate deficit
        if(deficit)
            Deathrate.decrease()
            Growthrate.increase()
        else
            Deathrate.increase()
            Growthrate.decrease()
        if(random(1,100) % 10 == 0)
            DrawCard()
        if(Addmonth(1)>12)                 #one Year finished
            populationHandler()            #add or remove people
            setYear(Year+1),setMonth(January) #reset Timer
        else
            Addmonth(1)
        if(Mode == 'challenge')
            checkWinConditions()
```

Main Loop - Pseudocode

Simulation

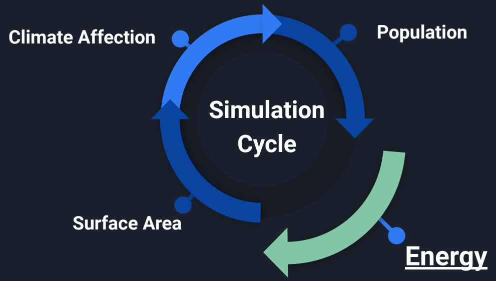


- Yearly increase/decrease

- Growth and Death rates depending on Energy Levels
    - Per Capita Consumption

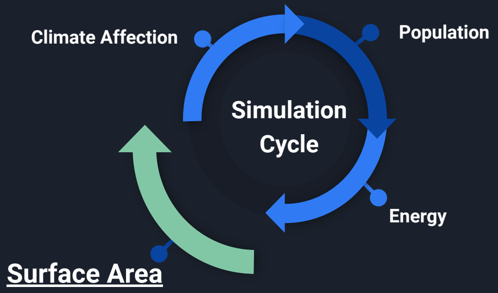
- Growth is main reason for City Growth

Simulation



- Energy is Resource
- Generated by Power Plants
  - Real-life orientated Plant Power Output
- Determined in each Cycle by the Energy Handler

Simulation

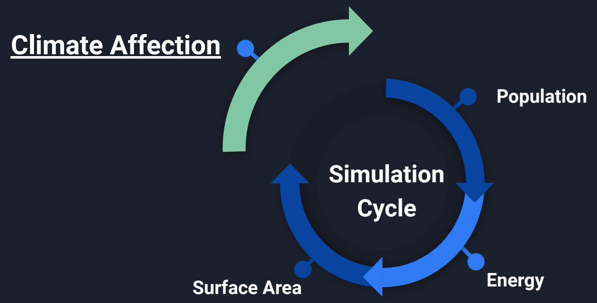


- Surface Area (Cells) is second Resource

- Limited

- Traded ´permanently´ for City growth or Power Plants

Simulation



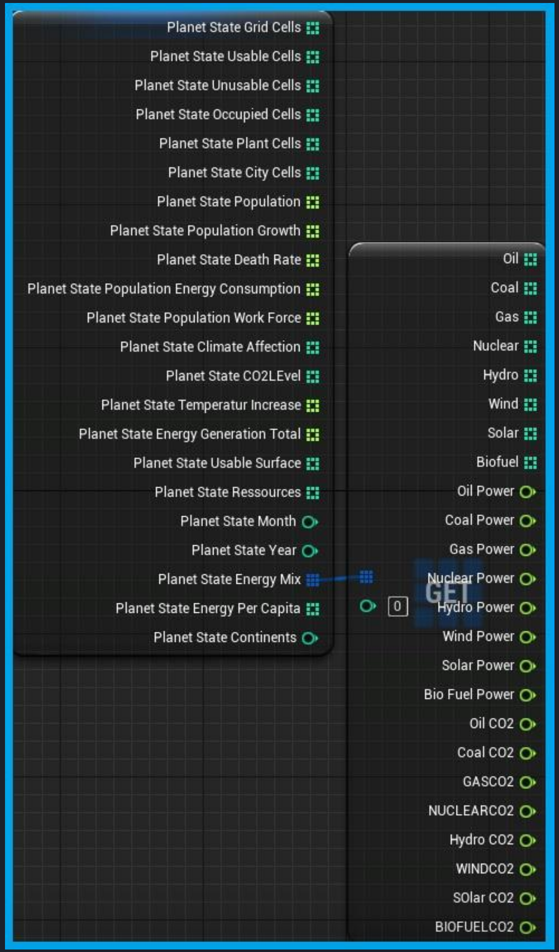
- Indicator for current Situation of the Planet

- Used for RNGs (Events)

- Level can be used for challenges

Simulation - Planet Handling

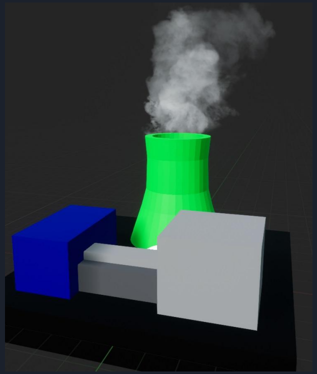
- Planet State Struct
    - Arrays to index Continents
        - Global and per Continent
        - Size = #Continents
    - Contains ´Energy Mix´ Struct
        - Holds Info about amount and type of Plants
    - Continuously updated
        - via Simulation Loop
        - via Event



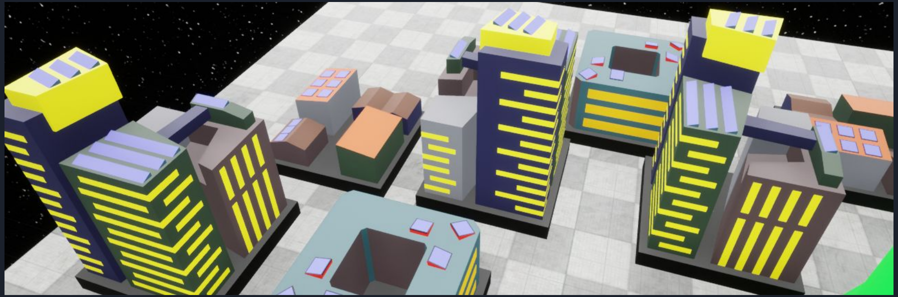
Interface and Events

- Global- and Per-Continent Bindings to distinguish each Continents Status
  - Separate Interface Elements
- Events will be triggered after Event Card is drawn (by Choice of Player)
  - Mali/Boni System
  - Only Way to get Points (Developments/Buildings) asides from Cheat Mode and Difficulty Setting
  - RNGs used, also based on Climate Affection
- Developments to manipulate Events and Spawnables
- Power Plants and Cities

Spawnables



- Power Plants
  - self-governing (e.g. after Spawn/Deactivation)
  - Building requires Points (Cost) and Time (Building Phase)
- Cities
  - Growth automatically
  - Initial Center selected



- Events (Movables)
  - Tornados
  - Floods

Demonstration

Discussion

Discussion

- Continent Creation:
  - Hashmaps

- Simulation
  - Cities:
    - Different grow patterns
  - Timer vs. While Loop

Sources

[1] A.F. Becker https://media04.lokalkompass.de/article/2019/03/08/0/10077630_XXL.jpg?1561900089
[2] imago/Christian Mang
https://www.tagesspiegel.de/images/fridaysforfuture-berlin-fridaysforfuture-schulstreik-fuer-klimaschutz-nach-veranstalterangaben-ueb/23964660/1-format43.jpg

[3] https://ourworldindata.org/world-population-growth

[4] https://ourworldindata.org/energy-production-consumption

[5]
https://ourworldindata.org/grapher/carbon-dioxide-emissions-factor?country=Shale+Oil~Charcoal~Diesel+Oil~Natural+Gas~Biogasoline~Biodiesels~Crude+Oil~Sub-Bituminous+Coal

[6]Source: http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html

[7] https://www.youtube.com/watch?v=QN39W020LqU

[8] Wikipedia Ray-Sphere Intersection
https://en.wikipedia.org/wiki/Line%E2%80%93sphere_intersection#/media/File:Line-Sphere_Intersection_Cropped.png

[6]Source: http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html

Beamer