

Simulating Pleistocene Park

New Mexico

Supercomputing Challenge

Final Report

April 11th, 2021

*Team 15*

*Monte del Sol Charter School*

*Team Members:*

*Brooklynn Martinez*

*Isabella Martinez*

*Kaley Martinez*

*Project Mentors: Charles*

*Strauss*

*Joann Mudge*

## **Table of Contents**

Executive Summary.....	3
Research and Context.....	4-5
Solving the Problem.....	5-6
Results .....	6-9
In the Future.....	9
Conclusion and Acknowledgments .....	10
References.....	11
Appendix A.....	12-18
Appendix B.....	19
Appendix C.....	20-21

**Executive summary:**

Climate change: a global phenomenon caused by an increase of carbon in our atmosphere that causes the Earth's climate to change. In Siberia, a project founded by Sergey Zimov, called the Pleistocene Park, works with megafauna in order to develop, reverse and prevent the permafrost currently melting in the Arctic. During the Pleistocene Period, megafauna such as mammoths existed and inhabited the environment there. The Pleistocene Project works to re-introduce similar animals in order to de-insulate the snow and revive the grasslands.

*Keywords:* snow patches, de-insulation megafauna, compact, climate change, global warming, Arctic Winter, simulation

## **Research and Context:**

Climate Change is a global phenomenon with limited time and limited solutions. With a little over a decade to relieve the extensive CO<sub>2</sub> influx in the atmosphere, the hunt for an answer is more critical than ever. Solutions, such as the *Abandoned Farmland Restorations and Efficient Aviation*, explained by Paul Hawkin, an environmentalist, have been implemented in order to stall the ramifications of global warming, but not to stop it. The time to find a solution with substantial consequences is now. Most do not understand that “now” existed decades ago. Back in the '80s, Sergey Zimov, a genius ecologist began the journey known as the Pleistocene Park Project Experiment. This project is one where megafauna, such as ox-bison, are used to mitigate climate change. As this project gained momentum, the “Russian government gave 144 km<sup>2</sup> of land for the experiment, and an official company was registered” (*Pleistocene Park Foundation*) nearly a decade later.

The Pleistocene Park project used aspects from the Pleistocene period ([Quaternary](#): 2.58 million years ago - 0.012 million years ago), pre-ice age. Its core is based on an ecosystem that existed in the Arctic during that time: the grasslands.

During the Pleistocene period, herbivores, such as mammoths, inhabited the grasslands. But because the grasslands were frozen over and buried by the Ice Age, an abundance of plant fossils and fumes (carbon dioxide) were locked underground, unable to decompose. The only thing keeping those old fumes locked away is the permafrost. The more severe global warming continues, the more the snow in the Arctic (and other permafrost regions) insulates, which inevitably thaws and melts permafrost.

Currently, the permafrost is acting as a “vault” (Paul Hawkin, 2017), keeping the toxic gases locked under miles upon miles of frozen soil and rock. As soon as those gases are released into the atmosphere, the additional Carbon Dioxide (and other toxic fumes) will teeter our already unstable atmosphere. To put it into perspective, there are currently “1.4 trillion tons of carbon estimated to be within the Arctic permafrost.” (*Pleistocene Park Foundation*, 2020) That is about “three times the amount of carbon inside the Earth’s forests (total)” and “two times the amount currently in the Earth’s atmosphere.” (*Pleistocene Park Foundation*, 2020)

In order to combat the thawing permafrost, the Pleistocene Project uses grazers and megafauna to compact and “de-insulate” the insulated snow. Allowing the reindeer, wolves, musk-ox, etc... stomp on the snow will release the insulation trapped inside. The goal in this is to expose the permafrost to the Arctic winters (which can be up to nine months long) by using animals to make the snow thinner and less dense. What this means, is by ridding regions of snow, the permafrost can now be directly impacted by the cool winter temperatures rather than the snow’s insulation. And now, because the snow is no longer an obstacle, a new grassland ecosystem can exist. Grasslands are known to reflect more sun off its grasses than the snow was ever able to (known as the Albedo Effect).

### **Solving the “Problem”:**

*What's the required megafauna needed in order to de-insulate the blanket of snow, in square kilometers (patches), covering the permafrost in the Arctic?*

The Pleistocene Park itself is an environment. And in this environment, in the Arctic, we need to somehow introduce millions of animals to begin renewing the grasslands. But how many animals do we need? What types of animals? And where can the animals be placed within the Arctic-permafrost regions? And will the regions in the Arctic adapt or collapse during the reintroduction of millions, to billions of animals?

Taking into account that the Pleistocene Park and the Arctic itself is an environment, we decided to create a simulation to model it. Unsurprisingly, all environments are complex and nuanced. It is important that our simulation is not only simple but realistic. This led us to focusing on one problem at a time: firstly, how many megafauna are needed to begin the renewal of the grasslands? In the Pleistocene Park Period era, bison, wolves, reindeer, and ox-musk and mammoths existed in the Arctic. For our simulation we decided to use a similar set-up (minus the extinct species).

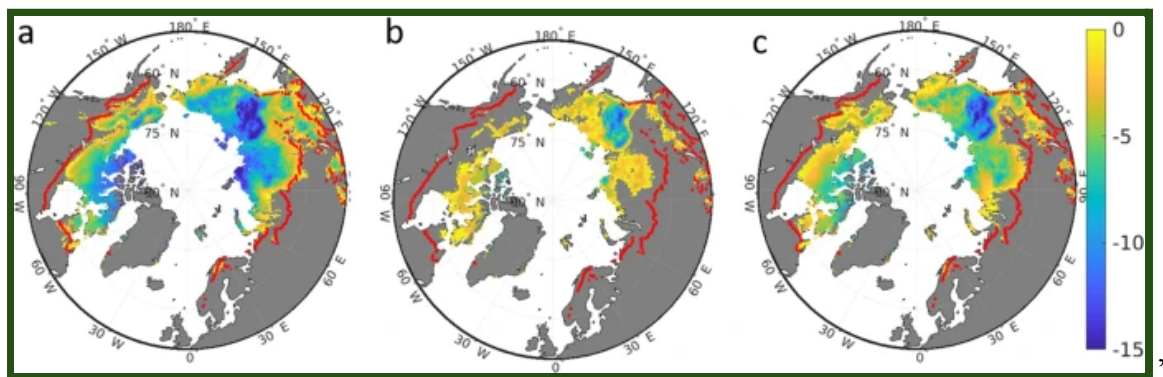
These animals will de-insulate the snow blanket that covers the permafrost, by stomping on and compacting it. Each animal living in today's Pleistocene Park de-insulate snow at

different rates. And, after trying and failing to find the rates at which each animal de-insulates snow, we decided to take on a new perspective. Instead, we used an animals-to-square kilometer ratio. So, now we had how many, and which types of animals needed per each square km of snow. And after we figured out how many square km there was in the Arctic, we decided to ask which parts of the Arctic are mitigatable--able to execute the de-insulation process?

## Results

### **Figure 1**

*Model of Arctic's de-insulation over time*



*Note: This model does not belong to me, but Beer & C., Zimov (contributors for the Pleistocene Project)*

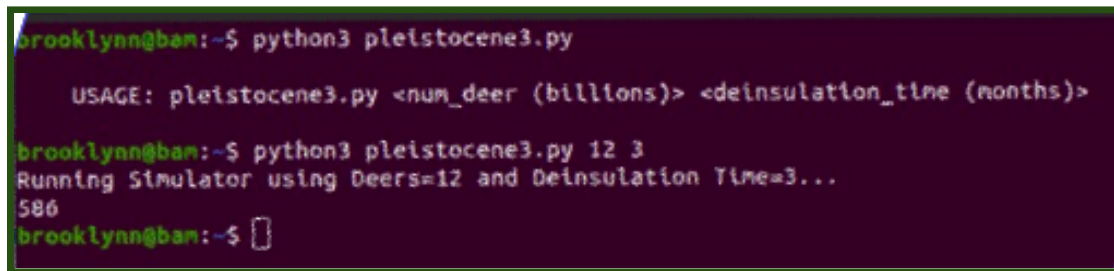
In the Arctic, some of the regions are too far gone to try to attempt any mitigation (permafrost has already begun thawing or is melted) and some regions do not need any mitigation (not affected enough by climate change). In *Figure 1* of the Arctic's de-insulation over time, -15°F to -10°F are the regions that do not need to be worried about right now. But -10°F to -5°F are the regions we want to mitigate as soon as possible. And -5°F to 0°F cannot be mitigated.

Through trial and error, we were left with three different simulations. They all performed similarly, but they looked at different factors individually. For example, our first simulation was

just simple Python code. But, eventually, we decided to study SimPy, a simulation-based framework that looks at an environment as a generator that produces results with specific inputs. We have two simulations, or generators, that use SimPy. Our first simulation inputs the snow in need of de-insulating, and using that number, it calculates the amount of deer alone it would take to fully mitigate that region. There were some limitations to this first simulation, seeing as it outputted an unrealistic or unreachable amount of reindeer needed in order for this to work.

## **Figure 2**

### *Simulation #1: Reindeer-only*

A terminal window with a dark purple background and green text. The prompt is 'brooklynn@bam:~\$'. The first command is 'python3 pleistocene3.py', which shows a usage message: 'USAGE: pleistocene3.py <num\_deer (billions)> <deinsulation\_time (months)>'. The second command is 'python3 pleistocene3.py 12 3', which outputs 'Running Simulator using Deers=12 and Deinsulation Time=3...' followed by the number '586'. The prompt returns to 'brooklynn@bam:~\$' with a cursor.

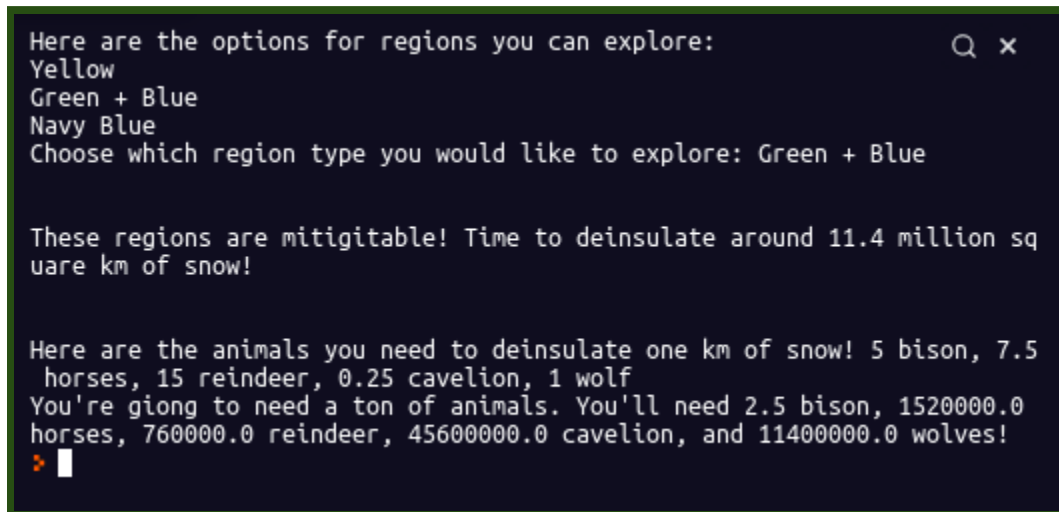
```
brooklynn@bam:~$ python3 pleistocene3.py
  USAGE: pleistocene3.py <num_deer (billions)> <deinsulation_time (months)>
brooklynn@bam:~$ python3 pleistocene3.py 12 3
Running Simulator using Deers=12 and Deinsulation Time=3...
586
brooklynn@bam:~$
```

In figure 2, we input the amount of snow in patches (mega hectares) and then we were given the amount of time it would take in months, and the amount of reindeer it takes to de-insulate. It takes 3 months for 137 billion reindeer to de-insulate ~927 patches (mega hectares) of snow.

With these results, we realized how unrealistic they were. Only implementing reindeer was far too burdensome on a single species, and impossible--also considering the reindeer only had a timespan of three months to do this. With this in mind, we decided to let *time* be something we input into *Simulation #3* (giving the animals a longer time period to de-insulate will be more effective).

### **Figure 3**

*Simulation #2: all animals, but no time limitations.*



```
Here are the options for regions you can explore:
Yellow
Green + Blue
Navy Blue
Choose which region type you would like to explore: Green + Blue

These regions are mitigatable! Time to deinsulate around 11.4 million square km of snow!

Here are the animals you need to deinsulate one km of snow! 5 bison, 7.5 horses, 15 reindeer, 0.25 cavelion, 1 wolf
You're going to need a ton of animals. You'll need 2.5 bison, 1520000.0 horses, 760000.0 reindeer, 45600000.0 cavelion, and 11400000.0 wolves!
> 
```

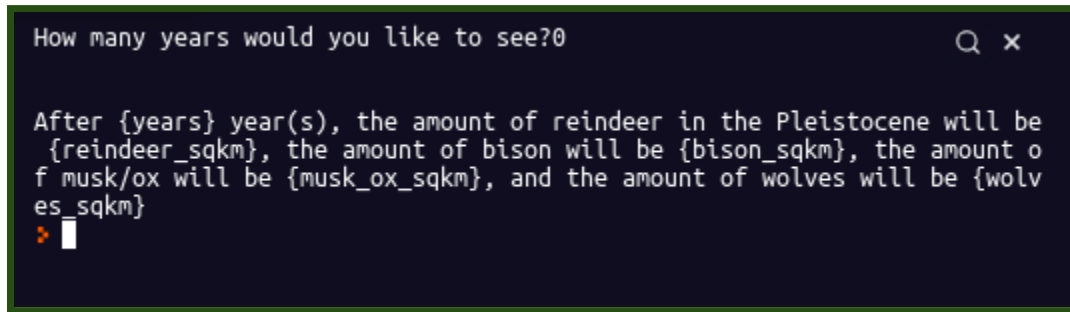
In *Figure 3*, this simulation is more user friendly, unlike the first one. We wanted to give the user some options and regions to explore, and in *Figure 3*, we chose to demonstrate the *Green + Blue* regions of the Arctic, which, if you remember, are the only regions mitigatable. It then outputs the amount of square kilometers of snow needed to be de-insulated. It then calculates the amount of animals needed to renew the grasslands in the Arctic. However, there is no reference to the amount of time it would take for this to happen. It just makes this de-insulation process happen in an instant, and that is unrealistic and impossible to execute in real-life.

Now we just need to let the user customize the amount of time to mitigate, and we'll be left with how many animals it takes to de-insulate a certain amount of snow in a certain amount of time. And this is how our third and final simulation was constructed.



## **Figure 4**

*Simulation #3: all animals, time limitations, birth and death rates.*

A terminal window with a dark background and light green text. The prompt 'How many years would you like to see?0' is at the top right. Below it, a multi-line output string contains curly braces for variables: 'After {years} year(s), the amount of reindeer in the Pleistocene will be {reindeer\_sqkm}, the amount of bison will be {bison\_sqkm}, the amount of musk/ox will be {musk\_ox\_sqkm}, and the amount of wolves will be {wolves\_sqkm}'. A cursor is visible at the end of the output string.

```
How many years would you like to see?0

After {years} year(s), the amount of reindeer in the Pleistocene will be
{reindeer_sqkm}, the amount of bison will be {bison_sqkm}, the amount of
musk/ox will be {musk_ox_sqkm}, and the amount of wolves will be {wolv
es_sqkm}
❖ |
```

In *Figure 3*, we implemented a customizable time frame, and birth and death rates for each animal. We wanted to try and implement as much reality as we could into our model, and we decided on birth and death rates. However, the birth and death rates annually for each animal are not inputted, and just used in the background for calculations. *Figure 3* demonstrates the blueprint of our final simulation. Someone is asked how many years they would like to mitigate in the only mitigatable regions in the Arctic (which are assumed to be the *Green + Blue regions*) and then it uses SimPy to loop through each year, killing and birthing the animals, and then finally outputting the final amount of each type of animals it will take to de-insulate the mitigatable regions in the Arctic, in a customizable amount of time.

## **In the Future:**

While simulating, it was obvious that many variables that occur in real-life environments, like natural disasters, or species interactions are possible and contribute to how an environment works. However, in our simulation of the Pleistocene Park, we were not able to account for every nuance. This is why simulations in general should be only seen as simulations, and possibilities, not as solid conclusions. We definitely want to try to add more visuals (like using Mesa in order to replicate an environment we can observe). So we would want to develop an agent-based-model or some of Mesa's aspects into our simulation. Also, looking into more

equation-based-modeling to see how we can further make our situation more complex (i.e. adding in the Albedo Effect equation:  $\alpha = (1 - D)\bar{\alpha}(\theta_i) + D\bar{\bar{\alpha}}$  ).

### **Conclusion:**

I think it is safe to infer that we have made various achievements throughout this span of time studying and working on the Pleistocene Park simulations. Through this journey we were able to be more in depth with our planet, and more importantly, how to save it. Sustainability is an extremely important part of mitigating climate change, whether we as humans decide to do that by condemning the meat industry and becoming vegan, to recycling, to building a simulation to help students better understand global warming in the Arctic...it is important to at least start. To start learning, to start helping, to start changing our lifestyles to live sustainably. I think that is the most important lesson we learned. We need to stop taking Earth for granted, especially at the pace we are deteriorating it at.

### **Acknowledgements:**

We could not have been able to educate ourselves on the Pleistocene Park Project without the support and patience of our mentors, Joann Mudge and Charles Strauss and our computer science teacher Rhonda Crespo. We have been working on this project in various different stages of the pandemic, and we are grateful we were able to retain and learn as much as we have about climate change and environmental science. We would also like to credit and thank Conner Cameron, a proficient python colleague of our mentor who helped clean, scrap, and solidify *Simulation #1* (Appendix A) which we referenced in Figure 2.

The Pleistocene Park Project needs all the support it can get! If you are interested in learning more about how you can assist ([donate](#), share and promote) them in the fight against global warming, make sure to check out the [Pleistocene Park Foundation](#) (located in the References)!

### **References:**

Anderson, R., (2017). Welcome to Pleistocene Park. The Atlantic.

[https://www.theatlantic.com/magazine/archive/2017/04/pleistocene-park/517779/?utm\\_source=share&utm\\_campaign=share](https://www.theatlantic.com/magazine/archive/2017/04/pleistocene-park/517779/?utm_source=share&utm_campaign=share)

Beer, C., et al. (2020). Protection of Permafrost Soils from Thawing by Increasing Herbivore Density. Sci Rep 10, 4170 doi.org/10.1038/s41598-020-60938-y

Hawkin, P., (2017). The Most Comprehensive Plan Ever Proposed to Reverse Global Warming and Abandoned Farmland Restoration. *Project Drawdown*.  
<https://www.drawdown.org/solutions/abandoned-farmland-restoration>

Macias-Fauria M, Jepson P, Zimov N, Malhi Y. 2020 Pleistocene Arctic megafaunal ecological engineering as a natural climate solution? Phil. Trans. R. Soc. B 375:

Macias-Fauria M, Pleistocene Arctic Megafaunal ecological engineering as a natural climate solution?

Nikita Zimov, Protection of Permafrost Soils from Thawing by Increasing Herbivore Density, available at:  
<https://www.nature.com/articles/s41598-020-60938-y#cities>

Regional Pleistocene ecoregions. Researching the effects of large herbivores on the arctic tundra/grasslands ecosystem. *Wikipedia*  
[https://en.wikipedia.org/wiki/Pleistocene\\_Park](https://en.wikipedia.org/wiki/Pleistocene_Park)

The Pleistocene Foundation:  
<https://pleistocenepark.org/>

**Appendix A: *Figure 2, Simulation #1: Reindeer-only***

```
# used emacs as an IDE
```

```
#!/usr/bin/env python
```

```
import sys
```

```
import simpy
```

```
import random
```

```
RANDOM_SEED = 42
```

```
NUM_DEERS = 6
```

```
GLOBAL_INT = 100000000
```

```
DEINSULATION_TIME = 5 # random number...
```

```
T_INTER = 3 # put a new deer into simulation every # months
```

```
SIM_TIME = 200 # sim time in minutes, changeable/random number
```

```
class Pleistocene(object):
```

```
    """the snow will have to request deer(s) ,
```

When the snow gets one, the snow can start the de-insulation process and wait for it to finish...

which takes DEINSULATION\_TIME units.

"""

```
def __init__(self, num_deers, deinsulation_time):
```

```
    self.env = simpy.Environment()
```

```
    self.timespans = []
```

```
    self.num_deers = num_deers
```

```
    self.deinsulation_time = deinsulation_time
```

```
    self.snow_value = 1200
```

```
    #self.introduction_interval = introduction_interval
```

```
    self.deer_limit = simpy.Resource(self.env, self.num_deers) # Initializes a simply resource  
with count num_deers
```

```
    #self.introduce_me = simpy.Resource(self.env, self.introduction_interval) # introduce more  
deer
```

```
#    self.deinsulation = self.deinsulation(snow)
```

```
    # ALL ADDITIONAL RESOURCES GO HERE
```

```
#    def deinsulate(self, snow):
```

```
    def deinsulate(self):
```

## *Pleistocene Park Simulations*

```
        """The deinsulation processes. It takes the snow processes and tries to insulate it...."""

        #print("Deers removed %d%% of %s's insulation." %

        #    (random.randint(0, 82), snow))

        yield self.env.timeout(random.randint(1, self.deinsulation_time))

#    yield self.env.timeout(random.randint(1, self.deinsulation_time))


#    def snow(self, env, name, pp):

#        """the snow in pleistocene Park(tech. it doesn't arrive there) and requests a deer. It then
#        starts being insulated, waits for it to finish then "leaves" but it actually completed so it isn't
#        deinsulated more...?"""

#        print('%s starts grazing at %.2f.' % (name, env.now))

#        with pp.deer.request() as request:

#            yield request

#            print('%s starts deinsulation at %.2f.' % (name, env.now))

#            yield env.process(pp.deinsulate(name))

#            print('%s finishes insulation at %.2f.' % (name, env.now))


#    def setup(self):

#        """create the ppe, start it off with an initial number of deers and keep sending them in
#        approx. every t_inter units/months/."""
```

### *Pleistocene Park Simulations*

```
# ppe = PPE(env, num_deers, deinsulation_time)

# for i in range(6):

#     self.env.process(snow, env, 'Snow %d' % i, ppe)

#     while True:

#         yield env.timeout(random.randint(t_inter - 1, t_inter + 1))

#         i += 1

#         env.process(snow(env, 'Snow %d' % i, ppe))

# print('Pleistocene Project Experiment')

# print('simulating...')

# random.seed(RANDOM_SEED)

# env = simpy.Environment()

# env.process(setup(env, NUM_DEERS, DEINSULATION_TIME, T_INTER))

# env.run(until=SIM_TIME)
```

```
def deinsulate(simulator):
```

```
    """Utilize deer_limit resource to remove snow"""
```

```
    rounds = simulator.env.now
```

```
    snow = simulator.snow_value
```

```
    with simulator.deer_limit.request() as request:
```

```
        yield request
```

## *Pleistocene Park Simulations*

```
        yield simulator.env.process(simulator.deinsulate())

# print(simulator.deinsulation_time)

# print(simulator.env.now)

# print(snow * float(1/simulator.env.now))

simulator.timespans.append(simulator.env.now)


def run_simulation(simulator):

    """Runs simulation"""

    for i in range(1200): # add some snow units

        simulator.env.process(deinsulate(simulator))


if __name__ == '__main__': # main runtime for simulator

    if len(sys.argv) < 3: # two required arguments from cl

        print(f'\n  USAGE: {sys.argv[0]} <num_deer (billions)> <deinsulation_time (months)>\n')

        sys.exit(1)

    num_deers = sys.argv[1] # set command line argument 1, 0 is the command itself

    deinsulation_time = sys.argv[2] # command line argument 2

    #introduction_interval = sys.argv[3] # command line argument 3
```



## *Pleistocene Park Simulations*

```
print(Running Simulator using Deers={num_deers} and Deinsulation
Time={deinsulation_time}...')

simulator = Pleistocene(int(num_deers), int(deinsulation_time)) # init main class

run_simulation(simulator)

simulator.env.run()

num_insulated = 0

for v in simulator.timespans:

    if v <= 6:

        num_insulated += 1

print(num_deinsulated+550)
```

**Appendix B: Figure 3, Simulation #2: all animals, but no time limitations.**

```
import simpy
import numpy

animals = '\nBison\nHorse\nElk\nReindeer\nWolf\nCavelion'
animlas_per_sq_unit = '5 bison, 7.5 horses, 15 reindeer, 0.25 cavelion, 1
wolf'

regions = '\nYellow\nGreen + Blue\nNavy Blue'
print(wHere are the options for regions you can explore: {regions}')

user_region = str(input("Choose which region type you would like to
explore: "))
print(f'\n')

if user_region == 'Yellow':
    print("\nSorry, it's too late to mitigate in the Yellow region\n because
the permafrost melted too much, temp is 0 to -5 degrees fahrenheit")
#yellow = 0 - -5 = temp of permafrost region
#for now, we are printing that it can't be mitigated, but we can change
that to something else if we Warning

elif user_region == 'Green + Blue':
    print('These regions are mitigatable! Time to insulate around 11.4
million square km of snow!')
    print(f'\n')

sq_km = 11400000

print(wHere are the animals you need to insulate one km of snow!
{animlas_per_sq_unit} ')
```

## *Pleistocene Park Simulations*

```
# calculate now how much animals you need for green region

#bison = 5 * 11400000
#horse = 7.5 * 11400000
#reindeer = 15 * 11400000
#cavelion = 0.25 * 11400000
#wolf = 1 * 11400000

# insulation rate * number of their animal = how much that type of animal
will deinslate
# 5 * x = 11400000, 11400000 / 5 = x
bison = 0.5 * 5 # the second number, in this case 5, is the insulation
rate per square km for each animal
horse = 11400000 / 7.5
reindeer = 11400000 / 15
cavelion = 11400000 / 0.25
wolf = 11400000 / 1

print(f"You're going to need a ton of animals. You'll need {bison} bison,
{horse} horses, {reindeer} reindeer, {cavelion} cavelion, and {wolf}
wolves!")
```

**Appendix C: Figure 4, Simulation #3: all animals, time limitations, birth and death rates.**

```
import simpy

# First, we design a process: which is the Pleistocene, duh. we put (env) in there because you
# have to have a reference to an Environment to run the process/generator later on:)
def ppp(park):

# in SimPy, we utilize infinite loops to create a never ending generator (of course, it ends when
# we tell it to, in the end)
    while True:
        reindeer_sqkm = 760,000
        bison_sqkm = 760,000
        musk_ox_sqkm = 760,000
        wolves_sqkm = 760,000
        print('starting at: %d' % park.now)
        years_duration = int(input('how many years would you like to see?'))

# these calculations are used to
        reindeer_birthrate = 100,000 * years_duration
        reindeer_deathrate = 139,000 * years_duration
        reindeer_deathrate = reindeer_deathrate
        reindeer_sqkm = reindeer_sqkm - reindeer_deathrate + reindeer_birthrate
```

## *Pleistocene Park Simulations*

```
bison_birthrate = 100,000 * years_duration
```

```
bison_deathrate = 139,000 * years_duration
```

```
bison_sqkm = bison_sqkm - bison_deathrate + bison_birthrate
```

```
musk_ox_birthrate = 100,000 * years_duration
```

```
musk_ox_deathrate = 139,000 * years_duration
```

```
musk_ox_sqkm = musk_ox_sqkm - musk_ox_deathrate + musk_ox_birthrate
```

```
wolves_birthrate = 100,000 * years_duration
```

```
wolves_deathrate = 139,000 * years_duration
```

```
wolves_sqkm = wolves_sqkm - wolves_deathrate + wolves_birthrate
```

```
print(f'after {years_duration} year(s), the amount of reindeer in the plectacine will be  
{reindeer_sqkm}, the amount of bison will be {bison_sqkm}, the amount of musk/ox will be  
{musk_ox_sqkm}, and the amount of wolves will be {wolves_sqkm}')
```

```
# this env.timeout is just saying, hey, im done going into the future the amount of years you  
wanted, the year duration is done, and the yield says 'oh, hey, let's move on maybe lol'
```

```
yield park.timeout(years_duration)
```

```
park = simpy.Environment()
```

```
# create an instance of this environment
```

```
park.process(ppp(park))
```

```
# now run this generator, and tell it to run until how many seconds.
```

```
park.run(until=100)
```

