Process System implementation

INTRODUCTION

The problem this implementation tries to solve is how much each process should transform input compounds into output compounds.

The idea of the implementation is to apply an economic model to the system, updating it on each frame, and using it to get the desired rate.

Each compound has then a demand, a supply, and a price, and the rate is calculated using them, and then the demand is calculated using said rate.

1: CALCULATING THE PRICES

At the beginning of the update method, each process calculates its price, and the price function it's a function of the demand, the supply, and the previous price of the compound. The supply of a compound is simply the amount of it the compound bag has, and the demand and old price are calculated on a previous call to the update method (and the initial values are arbitrary).

In the current implementation the function is

$$P = \sqrt{(D / (S + 1))} * COMPOUND\_PRICE\_MOMENTUM + oldP * (1 - COMPOUND\_PRICE\_MOMENTUM))$$

With:

- P the price
- D the demand
- S the supply
- oldP the old price
- COMPOUND_PRICE_MOMENTUM a constant between 0 and 1.

The function also performs a check to raise the value from 0 to a small positive number if needed.

After that is done, if the compound is marked as "useful" (this is done on the compound table), then the price gets inflated by a certain amount, depending on the compound supply. This is done so that the process system knows which compounds does it want to make (otherwise all prices would decay to 0 and that would be no fun).

In the current implementation, the price inflating function is

$$PI = IMPORTANT\_COMPOUND\_BIAS / (S + 1)$$

With:

- PI the price inflation
- S the supply
- IMPORTANT_COMPOUND_BIAS an arbitrary constant

It's important to notice that for the first price equation the old price used it's the non-inflated one.

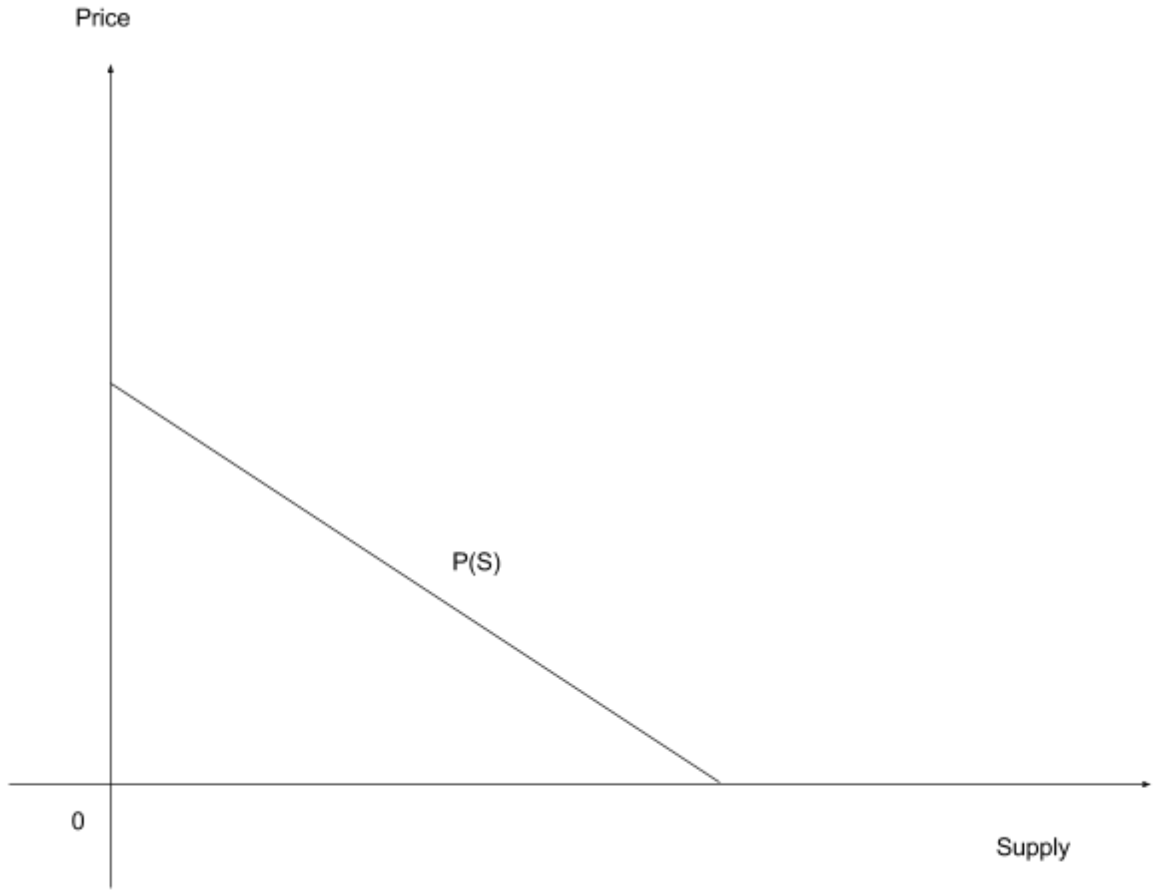
If, after all that, the price is below a certain small number, then the price is rounded down to zero (and therefore dumped later).

PHASE 2: CALCULATING THE RATE

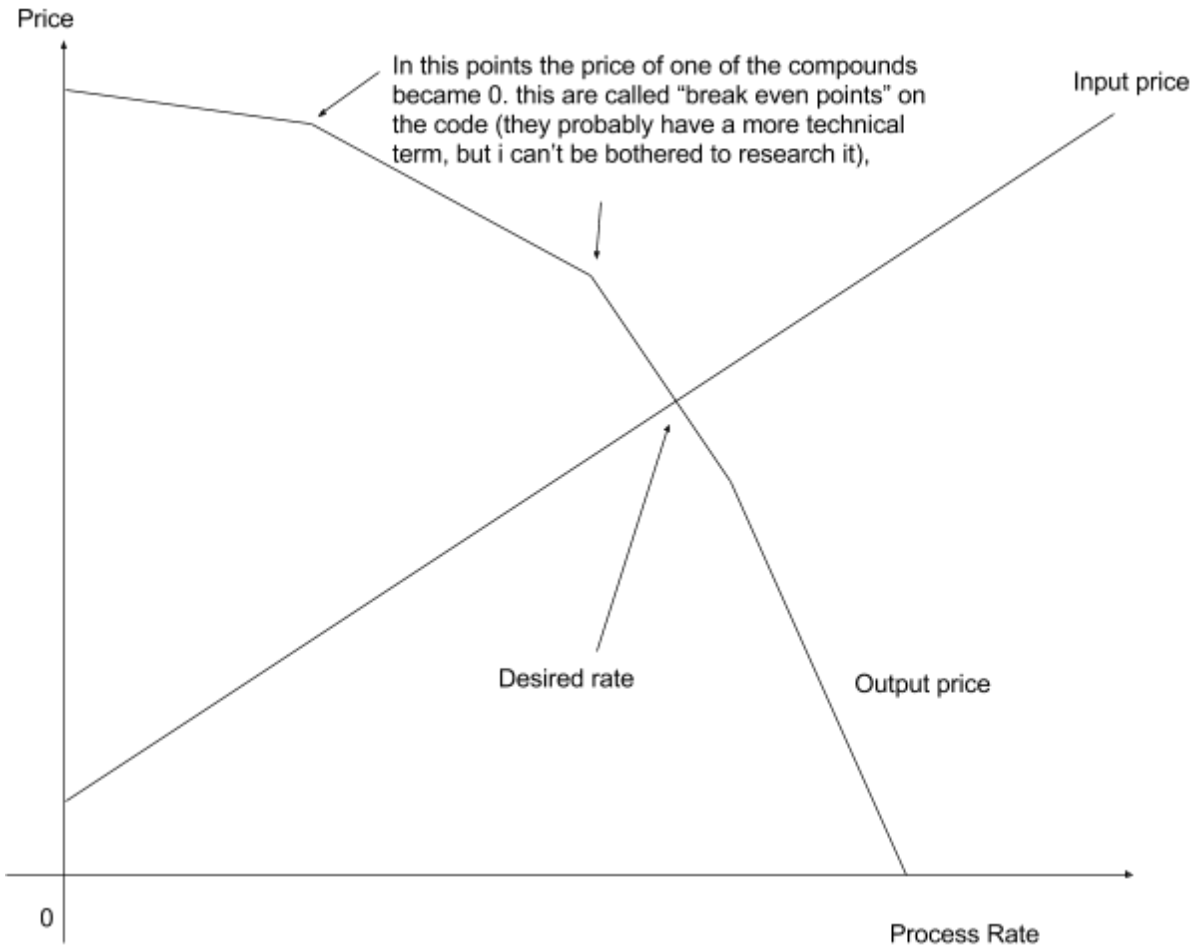
After all those calculations are done for all of the compounds, the process system needs to decide the desired rate for each of the processes.
This rate could, potentially, be larger than the maximum capacity the process has.

Said rate is calculated by making a simple prediction about the prices of the compounds, by replicating the same calculations of phase 1 on each compound, but assuming the supply is one unit larger, and then assuming the price is linear in respect of the supply.
If the price of the compound was 0 on phase 1 it's assumed that the price of that compound is always 0, regardless of the supply.
Finally, it's assumed that the price is 0 for any value of supply that the linearization would make it's price negative (aka, the minimum value of a function is 0).

Here's an example of one of those functions, made in the google drive equivalent of MS Paint:



Then all the process system has to do is get the rate at which the sum of all the input compound prices would equal the sum of all the output compound prices, and that is the desired rate.
If all of the output compounds have a price of 0, then the desired rate is made 0.
An example of a process would be like this:



To account for the storage space, a similar calculation is made but multiplying the prices of the compounds by a function that returns a value between 0.0 and 1.0, depending on the available space and the size and amount of the compounds processed.
In the current implementation, the function is:
M = 2.0 * (1.0 - sigmoid(RS / (AS + 1.0) * STORAGE_SPACE_MULTIPLIER));
With:

M is the multiplier between 0 and 1.
RS is the required space of the compound (volume * amount produced/processed)
AS is the available storage space
STORAGE_SPACE_MULTIPLIER is an arbitrary constant (currently 2.0)

Then the inputs get converted into the outputs by a rate of the minimum of the desired rate considering the storage space ONLY if by doing so the rate is reduced, otherwise the space is not considered (this is to avoid the process system to destroy compounds to gain space, when the compound purging code should do that), and the max capacity of the process (assuming there are enough inputs and storage space to do so).

PHASE 3: CALCULATING THE DEMAND

After the desired rates are found, the process must determine the demand of the compounds.

The demand of a compound is equal to the sum of all the demands generated by all the processes, which is defined by the equation
$D = DR * IN * soft(PC * IN)$
with:

D the demand generated by a process.

DR the desired rate of that process (without capacity, input or storage space limitations being considered).

IN the input needed by the process (aka the amount of input spent on running the process at a rate of 1).

PC the maximum process capacity

soft(x) a continuous, monotonically increasing function that equals 0 when $x = 0$, and 1 when $x \longrightarrow \infty$

In the current implementation
$soft(x) = 2 * sigmoid(x * PROCESS\_CAPACITY\_DEMAND\_MULTIPLIER) - 1$
with PROCESS_CAPACITY_DEMAND_MULTIPLIER being an arbitrary constant.

It's important to note that a process like $A \longrightarrow B$, with a rate of $2 * x$ it's exactly the same as the process $2 * A \longrightarrow 2 * B$, with a rate of x
It's also important that processes with a max capacity of 0 do not affect the system in any way.

That's it! :D