The Energy Market

Goal

The goal of this project is to design and implement a multi-process and multi-thread simulation of an Energy Market in Python.

Design

The design of the project will contain four main processes (each process can be modelled to fit the design intended by the creator).

- ***Home***: Energy producing and consuming home with initial rates as well as a specific trade policy (one of three): Always sell to the market, always give away, sell if no takers.
- ***Market***: Current energy price which will evolve according to transactions with home, weather conditions and random events. This process will be multi-threaded and will carry out transactions with home is separate threads. There will be a limit to how many simultaneous transactions can take place with homes.
- ***Weather***: Simulation of weather conditions which will impact energy consumption (temperature, storms, etc.).
- ***External***: Simulation of random events of specific types that can impact energy prices.

Each process will communicate with each other in a specific manner.

- Homes communicate with each other and the Market via message queues.
- Weather updates a shared memory with the specific weather conditions.
- External (**child** of Market), signals events to its parent whom will act accordingly on the energy price.
- Home and Market update the terminals they are connected to showing the progress of the simulation.

PSEUDO CODE

Home:

```
PROCESS home
Random consumption (float between 0 and 1)
Random production (float between 0 and 1)
Random trade_policy (int between 0 and 2)
Random renewable_energy (int between 0 and 2 (0 means normal home, 1 is solar, 2 is wind)
WHILE TRUE
    IF renewable_energy isn't 0
        Multiply production by specific coefficient depending on SEASON // Wind works better in Autumn
    END_IF
    Determine the difference between production and consumption
    Put (difference, trade_policy) in message_queue
END_WHILE
```

Weather:

```
PROCESS weather
Global random temperature (float between 0 and 1 where 0.5 = 0°C)
Global season (int between 0 and 3; Spring, Summer, Autumn, Winter)
Update shared_memory with temperature and season
```

Market:

```
PROCESS market
Get external_event signals from external and determine the associated coefficients
Get energy_price stored previously
IF home processes are running
    Wait before starting
WHILE TRUE
    Set overall_production to 0 // Will store the cumulated difference in message_queue depending on trade_policy
    Put (difference, trade_policy) tuple from message_queue into home_list
    Create 3 variables; all_market_energy, all_home_energy, market_if_no_home_energy
    FOR (difference, trade_policy) in home_list
        IF difference is negative or null
            Add difference to overall_production
        ELSE
            SWITCH trade_policy
                CASE "always sell to market"
                    Add difference to all_market_energy
                CASE "always give away"
                    Add difference to all_home_energy
                CASE "sell if no takers"
                    Add difference to market_if_no_home_energy
            END_SWITCH
        END_IF
    END_FOR
```

```
        IF all_home_energy + overall_production positive or null
                overall_production = 0
                Add market_if_no_home_energy and all_market_energy to
                overall_production
        ELSE
                Add all_home_energy, all_market_energy and
                market_if_no_home_energy to overall_production
        END_IF
        Determine current energy price from data and save it to energy_price
END_WHILE
```

External:

```
PROCESS external
Select 2 random signals (from a chose list of signals)
Send both signals to market
```

Implementation Plan

Firstly, we will create our home processes and make sure that they talk to each other correctly with hard coded production and consumption values and trade policies to make sure that the final production value (positive for producing more than consuming, negative otherwise) is correct.

Secondly, we will create the market process and verify that the data it receives from home processes corresponds to what is sent and make sure that each home is connected to the market by a single thread. It should receive positive or negative values from each home depending on their consumption, production rates and trade policies.

We can then work on implementing the weather and external processes individually and making sure the shared memory can be accessed by all processes (as a shared memory should) and modified by weather only. We will also make sure that external can send signals to market and that market can interpret them properly.

To automate the start up, we could give the python code a given number of home processes to start with and have it set up all the proper processes alone and make sure that market and home send the required information to the terminal when needed (we could use a keyboard signal to demand specific information).

To free up the resources, we can empty all message queues after we want to close the simulation by a SIGKILL or something else and close all threads and processes before exiting the simulation.