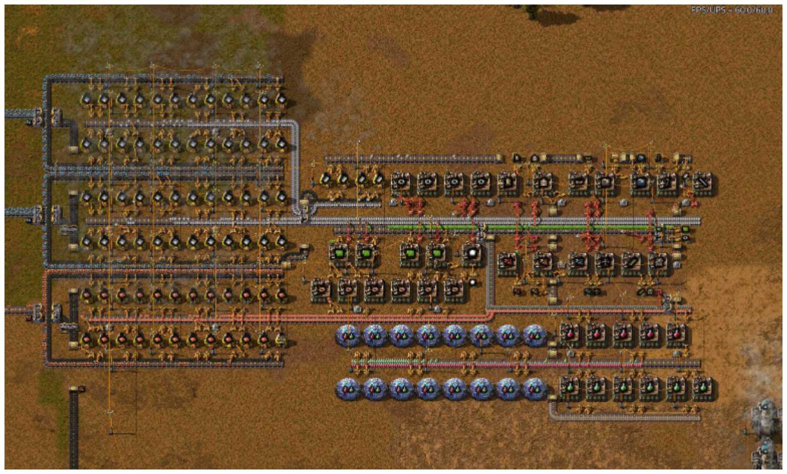
Figure 13: Factorio

27 Factorio Throughput Calculator

In this project, you will simulate a simplified Factorio style production line on a 2D grid of tiles. Each tile may be a resource source, conveyor belt, splitter, assembling station, or chest. The system advances in ticks. At each tick, tiles may generate resources, move items, process recipes, or collect output. The goal is to calculate how many ticks the blueprint produces resources and how many final items are produced before the system becomes stable or clogged.

The simulation is considered finished when, for 5 consecutive ticks, there is no change in the global state (no items move, no items are produced or consumed, and no tile inventory changes).

27.1 Tile Types

The following tile types are available:

Source:

- Generates 1 resource per tick of a fixed type (e.g. copper or iron).
- Holds up to X items internally.
- Attempts to push one item each tick to its output neighbor if that neighbor can accept an item.

Conveyor belt:

- Straight belt with a fixed facing direction.
- Can hold at most one item.
- Each tick, tries to push its item to the next tile in its facing direction if that tile can accept an item.

Splitter:

- T shaped tile with one input side and two output sides (left and right).
- Can hold at most one item.
- Each tick, when pushing an item, alternates between left and right output.
- If one output is blocked, it sends everything to the other output if possible.

Assembling station 1:

- Holds up to 2 items.
- If it contains at least 2 copper items, after 2 ticks of processing it consumes 2 copper and produces 1 copper wire.
- Attempts to push item to neighbor if that neighbor can accept an item. (you can optionally fix the output to one direction)

Assembling station 2:

- Holds up to 2 items.
- If it contains at least 1 iron and 1 copper wire, after 3 ticks of processing it consumes those 2 items and produces 1 inductor.
- Attempts to push item to neighbor if that neighbor can accept an item. (you can optionally fix the output to one direction)

Chest:

- Counts and stores all items inserted.
- Used to measure total produced output.

You may choose reasonable item names and internal limits where not explicitly specified, as long as they are clearly documented.

27.2 Input Format

The program reads the blueprint from a text file, for example `blueprint.txt`, with the following template:

```
ROWS COLUMNS
<row 1>
<row 2>
...
<row ROWS>
```

Each row is a string of exactly `COLUMNS` tiles separated by space. Each string slice represents one tile on the grid. For example:

```
7 6
. . SI100 . . .
. . BD . . .
. . BD . C .
. . BR BR TL .
. . . . 1 .
. . . . BD .
. . . . C .
```

You must define and document a legend mapping characters to tile types and orientations, for example:

`.`	empty tile
`SCX`	copper source with capacity X
`SIX`	iron source with capacity X
`BR`	right facing belt, `BL` left, `BU` up, `BD` down
`TL`	splitter input form left, `TR` input from right, `TU` input from up, `TD` down
`1`	assembling station 1
`2`	assembling station 2
`C`	chest

Optionally, you may allow a separate configuration file for parameters like bu er sizes or recipe times, but a single blueprint file is sufficient.

27.3 Implementation Guidelines

1. **Running the Program**:

   Parse the grid from the input file and construct an internal representation of tiles, their types, and directions.

   Implement a global tick based simulation. In each tick:

   - Sources generate items (if not empty) and attempt to push to their output neighbor.
   - Belts, splitters, and stations attempt to move or output items.
   - Assembling stations progress their internal timers when they have the required items.

   Ensure that all item moves for a tick are resolved consistently, without a tile overflowing the allowed item number.

   Detect the end of simulation when the global state does not change for 5 consecutive ticks.

   At the end, report:

   - Total number of ticks simulated.
   - Total number of items of each type stored in each chest.

   Render the blueprint using Swing

27.4 Testing

Create small blueprints that test individual components:

- Single source feeding a belt into a chest.
- Source into splitter into two chests.
- Minimal assembling line producing a single inductor.

Test longer production lines with multiple sources and shared belts. Measure:

- Total ticks until the system stabilizes.
- Items per tick entering each chest over time (throughput).

Create blueprints that intentionally clog the system, for example by having too few belts or no chest at the end, and verify that the simulation stops with zero changes over 5 ticks.

Compare blueprints with di erent layouts for the same logical production (for example straight line vs split and merge) and discuss their throughput di erences.

compare simulation times for di erent sized blueprints