**Batch Processing – Modular production line game**

Game overview

The player is provided with a set of modular tools, on each level, the player must use these tools to invent a production line that creates the desired output.

Core game loop

* Player receives output from a wall
* Player places modular tools to build production line
* Player tests line, iterates and makes changes until the output is successful
* Optionally, player may choose to optimize their solution

Game Challenges

The biggest challenge of the game is to invent your own solution. You are given several tools and shown how they work, now it is up to you to create a solution that works.

There are however a few further challenges:

**Space**

Such as in games Infinifactory and SpaceChem, you may create an inelegant solution to a problem. Like in programming, you may come to a stage later in the game where you realise you need to do something you have already done before, and so you can essentially copy and paste your previous solution, however you find that it takes up too much space and simply will not fit within the boundaries your production line is required to fit in, and thus the solution requires optimisation.

Example: *See fig 1.*

**Profit and Cost of resources**

Games such as Big Pharma impose a resource cost onto the tools of a production line; one of the challenges in Big Pharma is for your production line to produce as much profit as possible. Whenever you purchase a modular part of a production line in Big Pharma there is a cost associated with purchasing this tool, as well as a cost for running each machine (e.g. a machine may have a maintenance cost of £10 per every 100 cycles). This challenges the player to try to find an efficient solution that utilizes as little resources as possible and so this challenges the player to try and optimize their implementation.

Example: *See fig 1.*

**Cycle efficiency / time**

Cycle efficiency, or rather, the time it takes from input to output, once again comes down to the optimization of your level implementation. You may implement a solution that takes 15 cycles to finish whilst there may be a solution that requires only 9. This allows the player to replay the level and find a more efficient solution.

For example: in the context of a restaurant, you may need to create a solution that can output a meal once every 20 seconds. If your solution takes 30 seconds, then you need to look at how you can improve your solution.

*Fig 1. Unoptimized solution (left) with a space of 7x5 and resource cost of 23 versus an optimized solution (right) with a space of 5x5 and a recourse cost of 12.*

 

Mechanics

**Conveyer belt – 1 cycle**

Moves food between locations

**Heater – added onto conveyer belt**

Keeps food warm as it passes

**Chopper – 2 cycles**

Chops up food

**Mixer – 2 cycles**

Mixes two or more ingredients together

**Boil – 2 cycles**

Boils food e.g. eggs

**Fridge – 2 cycles**

Cools food or keeps food cool

**Masher – 2 cycles**

Mashes food, e.g. potato

**Oven / Grill – 3 cycles**

Cooks food

Example problem and solution

*See ExampleLevel.png*

Bibliography

Infinifactory: <http://www.zachtronics.com/infinifactory/>

SpaceChem: <http://www.zachtronics.com/spacechem/>

Poly Bridge: <http://polybridge.drycactus.com/>

Brown, M. (2015) *Puzzle Solving… or Problem Solving? | Game Maker’s Toolkit*. [Video] Available online: <https://www.youtube.com/watch?v=w1_zmx-wU0U> [Date of access: 05 October 2017]