POWERS OF TEN

*Foster **Ideation***

PREREQUESITES

- Mathematical knowledge about powers

TIME

5-50min

WHAT

The method Powers of Ten is a reframing technique based on the short film of the same name from the year 1977. The movie shows a picnic scene in Chicago with different magnitudes of distance from 1 meter up to 10^{24} meters and then down to 10^{-16} meters. It takes you up into the deep universe and then back into the hand of the man that is at the picnic until you only see atoms.

WHY

The first interesting thing about Powers of Ten is that you don't have linear, but **exponential growth**. For humans it is hard to think exponentially because our brains are trained to think linearly. But a lot of things actually grow exponentially.
The second thing is changing the **point of view** - you can look at a problem from very different magnitudes of a certain dimension. Varying the point of view allows you to find the **right framing** for different issues of this problem - basically by zooming in and out the current scene.
When working on a virtual or design problem one needs to construct the whole model in one's mind to be able to work on it. This is a very complex process where Powers of

POWERS OF TEN

*Foster **Ideation***

Ten can help by first discovering the whole problem space with different magnitudes and then finding and **developing patterns**.

HOW TO

Powers of Ten is an exercise that allows to change the point of view by varying magnitudes of context. You can start at 10^0 and then work your way up and down.

FIND THE RIGHT SCALE The original short film is working with magnitudes of distance, but feel free to use any other scale that might fit in. Apart from the obvious ones like 'distance', 'size' and 'time' you could use others like 'cost', 'complexity' or even something abstract like 'reward points'.

MOVE IN AND OUT Step back and try to really understand how the problem that you are working on would fit into its context. Remember to move in Powers of Ten, so always add or remove a zero at the end.

DETECT PATTERNS By moving through the magnitudes of your scale try to find patterns that repeat at different magnitudes and try to group the information properly.

EXAMPLE

You can apply powers of ten in problem solving, insight development and ideation.

#1
In problem solving you can take the problem and reframe it in other magnitudes of context - try to define it in as many magnitudes as you can to get a better overview of your problem.

Imagine you get assigned a new ticket to fix a specific bug. It can help a lot to view the different issues of the problem by putting them into the defined scale.

Example scale to use:

> Product Experience > Product > Sprint > Programming Language > Framework > Software Architecture > Module > Class > Method > Block > Line of Code > Expression > Character > Bit > Electricity > Atom

Now while fixing the problem you can try to see it from different point of views, e.g.:

- How does the error appear in the product?
- In which sprint was the feature developed that is now broken?
- Does the framework help us to bypass the error?
- In which class is the bug?
- In which function?
- Is there maybe a 'hidden character' that breaks the code?

POWERS OF TEN

*Foster **Ideation***

#2
In insight development you can empathise with the user through different magnitudes of context.

Imagine you are designing a checkout experience. For developing new insights, you can try to think what happens if the user buys products of different values.

Example scale to use:

House (100000$) > Car (10000$) > Laptop (1000$) > Shoes (100$) > Pizza (10$) > Soft Drink (1$)

#3
In an ideation session creativity sometimes gets stuck and you are out of ideas. You can try to view the context from different magnitudes to explore the solution space or restrict your ideas in a certain magnitude of context.

Imagine you are a developing a new feature for a software. You could put specific constraints on your ideation to facilitate the flow of new ideas, e.g.:

- Endless time for development / Must be ready in one hour
- It can run on a supercomputer / It has to run on a low power computer
- The user has a whole day to complete the task / The user just has 10 milliseconds
- The UI is a big wall / There is no visible UI at all