VPS 5	Parallel and Distributed Software Systems	WT 21/22, Exercise 4

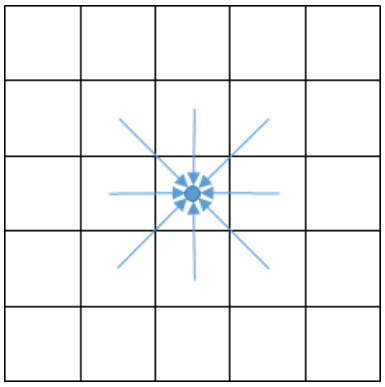
Deadline: 24.01.2022 15:30

Name Andreas Wenzelhuemer

Points _____ **Effort in hours** 5

1. Psychedelic Diffusions (4 + 4 + 4 Points)

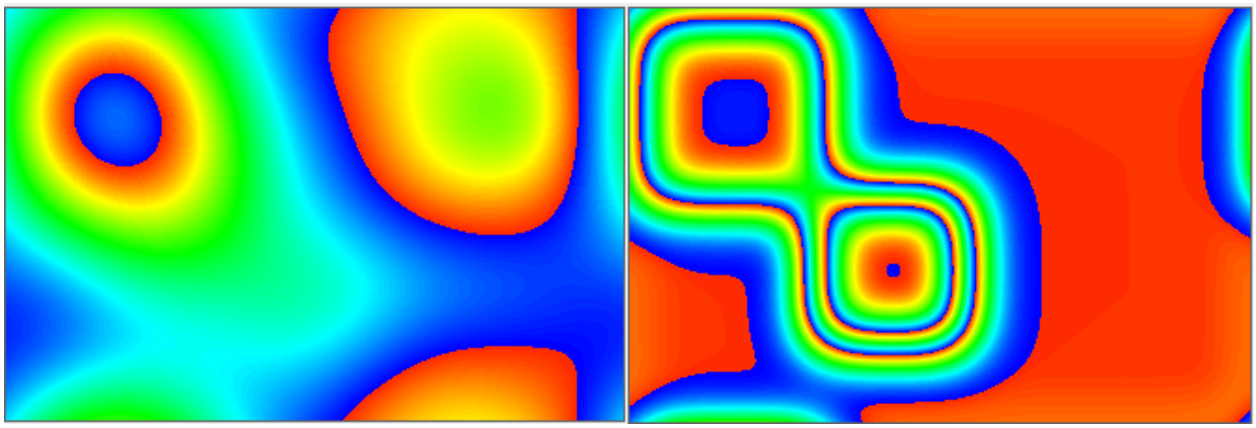
On Moodle you find a template for implementing a diffusion simulation. Simple diffusion simulations work by computing the value of a point by averaging over its neighboring points:



$$f(p) = \frac{1}{8} * \sum neighbor(p)$$

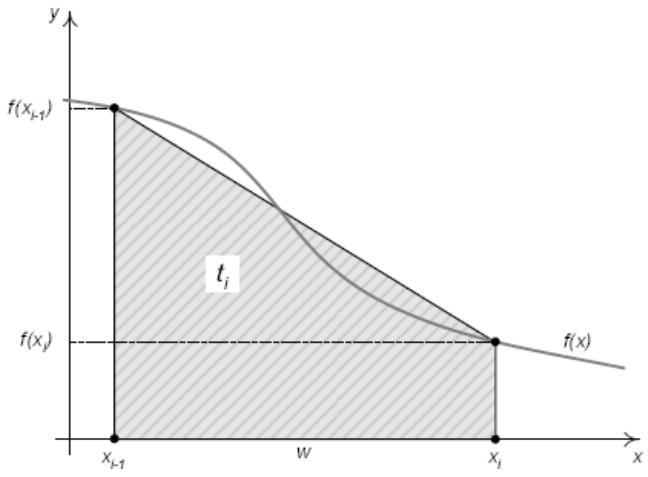
a) Complete the implementation of the simulation logic in C# using the provided template.
b) Use any of the learned techniques to compute the simulation in the background to provide a responsive UI. Take care of proper cancelation and locking!
c) Implement a parallel version of the simulation with a parallelization technique of your choice. Discuss your design considerations and calculate the speedup.

Document each step and also show a screenshot of the application in action.



2. Parallel Numerical Integration (5 + 5 + 2 Points)

Numerical integration is a technique for numerical (approximate) calculation of a definite integral of a function *f*. The algorithm is defined as follows: The interval of the integral is partitioned into many small parts of size *w*. For each of those small intervals the area under the curve is approximated by the area of the trapeze defined by x_{j-1}, x_j, $f(x_{j-1})$ and $f(x_j)$. Then the approximation of the whole integral can be computed by summing up all trapeze areas. The total accuracy of the calculation can be scaled arbitrarily by increasing the total number of trapezes.



As the calculation of one trapeze area is independent from the other areas, this parallel numerical integration algorithm can be parallelized quite easily.

a) Implement an OpenMP program in C or C++ for parallel numerical integration of the following function:

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

The user should be able to set the total number of trapezes with a parameter.

b) Implement another version of the program in C# using the .NET Task Parallel Library.

c) Test both versions and measure the consumed runtime for various configurations. Document your results in tabular form and analyze.

By the way, do you notice something concerning the value of the computed integral?