My first step was visualising the plate by modifying the code to use a 10x10 grid for rows and columns. Subsequently, I made several changes to ensure most of the computations run on the GPU device, even though this might have slowed the execution time.

I implemented three separate CUDA kernels to achieve this:

1. **Kernel 1:** Initializes the boundary conditions of the temperature array.
2. **Kernel 2:** Performs the core Laplace solving calculations.
3. **Kernel 3:** Computes the maximum temperature change (error) between the current and previous iterations and updates the previous temperature array accordingly.

The kernels include multiple if statements, likely contributing to the increased execution time. Since optimization wasn't a primary concern for this task, I focused on enhancing performance only a little. Although using three separate kernels was optional, I chose this approach for clarity and modularity. The main aim was to execute most of the CUDA device's code, ensuring parallel processing.

Here is a screenshot of the output after my modifications, showcasing the temperature distribution.