CS 3005: Programming in C++

Faster Calculations

Introduction

We want to be able to calculate our grid numbers more quickly. One way to accomplish this is to use more cores of the CPU. We'll use threads of execution to accomplish this task.

Assignment

In this assignment you will create a class `ThreadedGrid`, that inherits from the `NumberGrid` class and becomes the parent of the `ComplexFractal` class. The main purpose of the `ThreadedGrid` class is to override the `calculateAllNumbers` method so that it will use threads to calculate numbers simultaneously.

The `ppm_menu` program needs to adjust its commands.

The menu will now provide two options for calculation:

- `fractal-calculate`: Calculate the escape values for the fractal.
- `fractal-calculate-single-thread`: Calculate the escape values for the fractal, single-thread.

`fractal-calculate-single-thread` will call a new function to do the non-threaded version of calculation.

Programming Requirements

The following files must be updated or created and stored in the `src` directory of your repository.

Create `ThreadedGrid.{h,cpp}`

This class inherits from the `NumberGrid` class, and becomes the parent of `ComplexFractal`.

Data Members:

- A `std::vector` of task objects. You can decide what those task objects should be. You may even make a simple class to store the information for a single task. If so, that support class should be declared and implemented in the `ThreadedGrid.{h,cpp}` files.
- A `std::mutex` object to control critical access to the vector of task objects.

Methods:

- `ThreadedGrid();` The default constructor. Leaves the task queue empty, and lets the parent class default constructor set values for its data members.
- `ThreadedGrid(const int& height, const int& width);` Leaves the task queue empty, and passes the parameters on to the parent class constructor.
- `virtual ~ThreadedGrid();` Empty, but required destructor.
- `virtual void calculateAllNumbers();` Overrides the method of the parent class. Uses the `worker` method in several threads to do the work. Be sure to manage the task queue, and clean up the threads when they have finished. Use the thread library to decide how many threads to launch, based on the capacity of the computer. It may be useful to use `try/catch` when creating threads to protect against limited resources.
- `virtual void worker();` As long as tasks are available in the task queue, get one, and complete it. Should use `calculateNumber()` and `setNumber()`.

Update `ComplexFractal.{h,cpp}`

`ComplexFractal` should now inherit from `ThreadedGrid`. This will require minor changes to the class declaration in the header file and in the constructor chaining in the implementation file.

Update `image_menu.h` and `image_drawing.cpp`

Add this function:

- `void calculateFractalSingleThread(ActionData& action_data);` This function calls the `NumberGrid` version of `calculateAllNumbers` instead of the `ThreadedGrid` version, which would be the default. This is

accomplished with this syntax: `grid.NumberGrid::calculateAllNumbers()`, assuming `grid` is a reference to a polymorphic `NumberGrid` object.

Note that `calculateFractal()` will not change. However, because of `ThreadedGrid`'s override of `calculateAllNumbers()`, `calculateFractal()` will automatically use the threaded version.

Update Functions in `controllers.cpp`

- `void configureMenu( MenuData& menu_data )` add the new/updated actions with the names and descriptions listed below.

Table of New Commands

Command Name	Function Name	Description
fractal-calculate	calculateFractal	Calculate the escape values for the fractal.
fractal-calculate-single-thread	calculateFractalSingleThread	Calculate the escape values for the fractal, single-thread.

Update `Makefile`

The following commands should work correctly.

- `make hello` - builds the hello program
- `make questions_3` - builds the questions_3 program
- `make ascii_image` - builds the ascii_image program
- `make image_file` - builds the image_file program
- `make ppm_menu` - builds the image_file program
- `make all` - builds all programs
- `make` - builds all programs (same as `make all`)
- `make clean` - removes all .o files, and all executable programs

Additional Documentation

- C++ Reference
- Examples from class

Show Off Your Work

To receive credit for this assignment, you must

- use git to add, commit and push your solution to your repository for this class.
- successfully pass all unit tests and acceptance tests

Additionally, the program must build, run and give correct output.

Extra Challenges (Not Required)

- Create additional command(s) that let the user select how many threads to use.
- Override the `setPPM` method to be threaded.
- Are there other compute-resource bound parts of the code that take long enough to be worth the effort of converting to threaded?