Class name: MainWindow

Header:

#include <QMainWindow>

#include <QSettings>

#include <QSystemTrayIcon>

#include <QThread>

#include <QtCharts/QChartView>

#include <QtCharts/QLineSeries>

#include <QtCharts/QCategoryAxis>

#include <QtCharts/QPieSeries>

#include <QDateTimeAxis>

#include <QTimer>

#include <QLabel>

#include <QHBoxLayout>

#include <QLCDNumber>

#include <QtSql/QSqlDatabase>

#include <QDebug>

#include <QMessageBox>

#include <QMainWindow>

#include "minerprocess.h"

#include "nvapi.h"

#include "helppage.h"

#include "nvidiaapi.h"

#include "gpumonitor.h"

#include "structures.h"

#include "database.h"

#include "helper.h"

#include "nvocpage.h"

Friend class: GeneralTest

Inherits: QMainWindow

Name space: Ui

Description: Mainwindow class is the central controller of the program which will receive and responds most of the operations from user. Currently, it is mainly composed of 7 components including GPU and mining information overview (Overview page), system information (System page), miner controller (Mining page), GPU information details (Devices page), overclocking (Overclocking page) and project information (Help page). It will also store GPU and mining information to the database.

Key Private Variables:

|  |  |
| --- | --- |
| QList<QWidget \*> \* \_gpuInfoList | Holding the dynamic generated GPU information QWidget displayed in the Devices Page. |
| QList<QWidget \*> \* \_diskInfoList | Holding the dynamic generated disk information QWidget displayed in the Overview Page. |
| int \_deviceCount = 0; | The number of GPUs connected to computer. |
| int \_diskCount = 0; | The number of disks connecter to computer. |
| bool \_testing | Indicate whether the program is in test mode. |
| bool \_ui\_refresh\_enabled = true | Indicate whether the device and disk information refreshing on UI is enabled. |
| nvidiaAPI\* \_nvapi | Pointer to the NVIDIA GPU driver API. |
| Helper helper | Helper which provide some useful functions like reading files to QStringList. |
| Ui::MainWindow \*ui | The main window of the program. |
| MinerProcess\* \_process | The miner thread. |
| QSettings\* settings | The setting file (minerscoffee.ini). |
| QIcon\* \_icon | Pointer to the icon of the software. |
| QList<GPUInfo>\* \_gpusinfo | Buffer holding GPU information. |
| MiningInfo\* \_miningInfo; | Buffer holding mining information. |
| PoolInfo\* \_poolInfo; | Buffer holding pooling information |
| Database \* \_databaseProcess = nullptr; | The database thread. |
| HelpPage\* \_helpPage; | The Help page controller. |
| NvocPage\* \_nvocPage; | The overclocking page helper. |
| bool \_isMinerRunning; | Whether the miner has started or not. |
| bool \_isStartStoping; | Whether the miner has stopped or not after starting mining. |
| unsigned int \_errorCount; | Restart time of miner. |
| QSystemTrayIcon\* \_trayIcon; | Small icon displayed in the toolbar of computer. |
| QMenu\* \_trayIconMenu; | The mini menu after press the small icon in toolbar. |
| QAction\* \_minimizeAction; | Action of minimize main window. |
| QAction\* \_maximizeAction; | Action of maximize main window. |
| QAction\* \_restoreAction; | Action of restore main window. |
| QAction\* \_quitAction; | Action of quit main window. |
| QAction\* \_helpAction; | Action of open help page dialogue (not the one in main window). |
| QChart\* \_chart; | Hash rate chart. |
| QLineSeries\* \_series; | Line in hash rate chart displaying hash rate. |
| QLineSeries\* \_seriesBottom; | Bottom line of the shadows below the hash rate line. |
| QAreaSeries\* \_areaseriesHash; | Shadow below the hash rate line. |
| QDateTimeAxis \*\_axisX; | X-axis of hash rate chart. |
| QChart\* \_chartTemp; | Temperature rate chart. |
| QLineSeries\* \_seriesTemp; | Line in temperature rate chart displaying maximum temperature. |
| QLineSeries\* \_seriesTempBottom; | Bottom line of the shadows below the maximum temperature line. |
| QAreaSeries\* \_areaseriesTemp; | Shadow below the maximum temperature line. |
| QDateTimeAxis \*\_axisXTemp; | X-axis of temperature chart. |
| QChart\* \_chartHistory; | History information chart. |
| QList<QLineSeries \*> \_seriesHistory; | Lines in history information chart. |
| QDateTimeAxis \*\_axisXHistory; | X-axis of history information chart. |
| QChart\* \_tempPieChart; | Temperature pie chart. |
| QPieSeries \* \_tempPieSeries; | Series in temperature pie chart. |
| QList<QPieSlice \*> \* \_tempPieSlices; | Slices in the series of temperature pie chart. |
| QTimer \_hrChartTimer; | Timer to refresh hash rate chart. |
| QTimer \_tempChartTimer; | Timer to refresh temperature chart. |
| double \_currentHashRate = 0.0; | Current hash rate. |
| double \_maxChartHashRate = 0.0; | Maximum hash rate. |
| int \_plotsCntr = 0; | The center of the x-axis of hash rate chart. |
| double \_currentTempRate = 0.0; | Current temperature. |
| double \_maxChartTempRate = 0.0; | Maximum temperature. |
| int \_plotsCntrTemp = 0; | The center of the x-axis of temperature chart. |
| nvMonitorThrd\* \_nvMonitorThrd; | Thread monitoring NIVIDA GPUs. |
| Core\* \_current\_core = nullptr; | Current miner core. |
| Coin\* \_current\_coin = nullptr; | Current mining cions. |
| Pool\* \_current\_pool = nullptr; | Current mining pool. |
| float \_total\_hash\_rate = 0; | Total hash rate. |
| float \_est\_output\_usd = NAN; | Estimated output in USD. |
| float \_est\_output\_cny = 0; | Estimated output in CNY. |
| float \_est\_output\_coin = 0; | Estimated output in coins. |

Private Types:

None.

Constructor:

explicit **MainWindow**(bool testing = false, QWidget \*parent = 0):

Parameters: whether the main window is in test mode, parent widget.

Return: None.

Task: Initialize necessary variables of the class. Create and invoke all the threads. Initialized all the graphs and overclocking page and help page. Check and load the nvml library. Load the most recent settings of necessary check box, combo box from the settings files (MinersLamp.ini).

Destructor:

~***MainWindow***():

Parameters: None.

Return: None.

Task: Free necessary allocated variables and chart. Save the parameters of some important settings to setting file. Delete and terminate the threads.

Private Methods:

None.

Reimplemented Private Methods:

void **createActions**():

Parameters: None.

Return: None.

Task: Create actions including minimize, maximize, restore and close windows and open help dialog page (not the one in main window).

void **InitTray**():

Parameters: None.

Return: None.

Task: Create option buttons in menu of the icon in the toolbar and assign actions to them.

void **setupEditor**():

Parameters: None.

Return: None.

Task: Set fonts type, size and highlight in the log block in mining page.

void **setupToolTips**():

Parameters: None.

Return: None.

Task: Set tooltips for some components including the GPU power draw, temperature and memory clock.

void **loadParameters**():

Parameters: None.

Return: None.

Task: Load the most recent settings before last close to program including the miner path, miner argument, share only information in log, coin type, miner core, pool, wallet, worker of the program. These settings are stored in MinerLamp.ini.

void **saveParameters**():

Parameters: None.

Return: None.

Task: Save the current settings before close including the miner path, miner argument, share only information in log, coin type, miner core, pool, wallet, worker of the program. These settings are stored in MinerLamp.ini.

void **initializePieChart**():

Parameters: None.

Return: None.

Task: Initialize the pie chart and add slices to it.

void **initializeConstants**():

Parameters: None.

Return: None.

Task: add pools and core to cores map.

void **setLCDNumber**(QWidget \* widget, unsigned int value):

Parameters: LCD widget, value to change.

Return: None.

Task: Set the value of the given LCD.

void **plotGrapgh**(QString dateStart, QString dateEnd, int deviceNum):

Parameters: start and end date, target device number.

Return: None.

Task: Put the conditions into retrieve condition buffer and set the retrieve bit of database to make it draw a chart of history information.

void **isAllPromptVisable**(bool status):

Parameters: disable the prompt or not.

Return: None.

Task: display or disable the functions of each page tab.

void **applyOC**():

Parameters: None.

Return: None.

Task: apply GPU overclock settings to GPUs.

void **setComboIndex**(QComboBox \* comboBox, QString key):

Parameters: target combo box, the target value.

Return: None.

Task: Set the index of the target combo box to the one with the given value.

void **changeLabelColor**(QLabel \* label, QColor color):

Parameters: Target label, color.

Return: None.

Task: Change the color of the given label.

void **RefreshTempGraph**():

Parameters: None.

Return: None.

Task: Append the current temperature to the line chart.

Coin\* **AddCoin**(QString coin\_name):

Parameters: Name of the coin to add.

Return: Pointer to the newly created Coin.

Task: Create a new Coin object.

Core\* **AddCore**(QString core\_name, const QString& path, const QString& api, Coin\* coin, const QString& cmd):

Parameters: Name of the core to add; Relative path of the core; API URL of the core; Pointer to one supported Coin of the core; command line to use the core to mine the Coin specified by the fourth parameter.

Return: Pointer to the newly created / updated Core.

Task: Create a new Core with a supported Coin if the core name does not exist in the map; Update an existed Core with a new supported Coin if the core name exists in the map.

Core\* **AddCore**(QString core\_name, const QString& path, const QString& api, QString coin\_name, const QString& cmd):

Parameters: Name of the core to add; Relative path of the core; API URL of the core; Name of one supported Coin of the core; command line to use the core to mine the Coin specified by the fourth parameter.

Return: Pointer to the newly created / updated Core.

Task: Create a new Core with a supported Coin if the core name does not exist in the map; Update an existed Core with a new supported Coin if the core name exists in the map.

Pool\* **AddPool**(QString pool\_name, Coin\* coin, const QString& cmd):

Parameters: Name of the pool to add; Pointer to one supported Coin of the pool; command line to use a core to mine the Coin in the pool.

Return: Pointer to the newly created / updated Pool.

Task: Create a new Pool with a supported Coin if the Pool name does not exist in the map; Update an existed Pool with a new supported Coin if the core name exists in the map.

Pool\* **AddPool**(QString pool\_name, QString coin\_name, const QString& cmd):

Parameters: Name of the pool to add; Name of one supported Coin of the pool; command line to use a core to mine the Coin in the pool.

Return: Pointer to the newly created / updated Pool.

Task: Create a new Pool with a supported Coin if the Pool name does not exist in the map; Update an existed Pool with a new supported Coin if the core name exists in the map.

void **AddPoolsFromFile**(const QString& filename):

Parameters: Filename of the pool information file.

Return: None.

Task: Add pools automatically from a text file. The text file must in a specific format.

void **SetMiningArgs**():

Parameters: None.

Return: None.

Task: Set the mining arguments

void **StartMiningCore**():

Parameters: None.

Return: None.

Task: Start the mining core.

void **StopMiningCore**():

Parameters: None.

Return: None.

Task: Stop the mining core.

void **onMinerStarted**():

Parameters: None.

Return: None.

Task: Set the status of the miner to mining and apply the overclocking settings.

void **onMinerStoped**():

Parameters: None.

Return: None.

Task: Set the miner status to not running and clear mining data settings.

void **onHashrate**(QString& hashrate):

Parameters: Current hash rate.

Return: None.

Task: If given valid hash rate, the program will refresh the mining information and display them.

void **onError**():

Parameters: None.

Return: None.

Task: Add error count and make system prompt error.

void **onReceivedMiningInfo**(MiningInfo miningInfo):

Parameters: Mining information.

Return: None.

Task: Refresh buffer of the mining information to the given one.

void **onReceivedPoolInfo**(QList<PoolInfo> poolInfos):  
 Parameters: Pool information.

Return: None.

Task: Refresh buffer of the pools’ information to the given one.

void **refreshSystemSettings**():

Parameters: None.

Return: None.

Task: Refresh the virtual memory size of the system.

void **EstimateOutput**():

Parameters: None.

Return: None.

Task: Calculate and refresh the estimate output based on current hash rate and mean income of 24 hours of the pools.

const QColor **getTempColor**(unsigned int temp):

Parameters: Temperature.

Return: Corresponding color of the given temperature.

Task: Output the color of the given temperature which will be set to the display temperature labels’ color.

void **setPushButtonColor**(QPushButton\* pushButton, bool pressed):

Parameters: Target pushbutton and whether it is be pressed.

Return: None.

Task: Set the color of the pushbutton to make it feels like it has been pressed. This function is specially used to handle the pushbuttons of the page tabs.

void **updateSliders**(unsigned int gpu):

Parameters: GPU number.

Return: None.

Task: Update the sliders in the overclocking page of the given GPU.

void **saveConfig**():

Parameters: None.

Return: None.

Task: Save the configuration of the overclocking settings to MinerLamp.ini file.

Private Slots:

void **iconActivated**(QSystemTrayIcon::ActivationReason reason) :

Parameters: the activation reason of the tray icon.

Return: None.

Task: Under trigger or double click situation, the program will be active from hide mode.

void **on\_pushButton\_clicked**():

Parameters: None.

Return: None.

Task: Start or stop miner.

void **on\_checkBoxOnlyShare\_clicked**(bool checked):

Parameters: Checked status of the check box.

Return: None.

Task: Only display the share information if the check box is checked.

void **onReadyToStartMiner**():

Parameters: None.

Return: None.

Task: Restore the miner if it is ready.

void **on\_pushButtonSearchHistory\_clicked**():

Parameters: None.

Return: None.

Task: Request the program to display history information with given condition.

void **on\_dateTimeEditHistoryStartTime\_dateTimeChanged**(const QDateTime &datetime):

Parameters: date time the user entered.

Return: None.

Task: Show the display button if the start date time for searching has been changed.

void **on\_dateTimeEditHistoryEndTime\_dateTimeChanged**(const QDateTime &datetime):

Parameters: date time the user entered.

Return: None.

Task: Show the display button if the end date time for searching has been changed.

void **on\_spinBoxHistoryDeviceNum\_valueChanged**(int arg1):

Parameters: The index of device number the user entered.

Return: None.

Task: Show the display button if the device number for searching has been changed.

void **on\_pushButtonCancelAutoPage\_clicked**():

Parameters: None.

Return: None.

Task: Cancel automatic page size management.

void **on\_pushButtonChangePageSize\_clicked**():

Parameters: None.

Return: None.

Task: Set the virtual memory page size with given input.

void **on\_pushButtonMonitorPage\_Overview\_clicked**():  
 Parameters: None.

Return: None.

Task: The page inside the monitor tab to overview page.

void **on\_pushButtonMonitorPage\_Mining\_clicked**():

Parameters: None.

Return: None.

Task: The page inside the monitor tab to mining page.

void **on\_pushButtonMonitorPage\_System\_clicked**():  
 Parameters: None.

Return: None.

Task: The page inside the monitor tab to system page.

void **on\_pushButtonMonitorPage\_DevicesInfo\_clicked**():

Parameters: None.

Return: None.

Task: The page inside the monitor tab to devices page.

void **on\_pushButtonMonitorPage\_clicked**():  
 Parameters: None.

Return: None.

Task: Turn the outside page to monitor page.

void **on\_pushButtonOCPage\_clicked**():

Parameters: None.

Return: None.

Task: Turn the outside page to overclocking page.

void **on\_pushButtonHelpPage\_clicked**():

Parameters: None.

Return: None.

Task: Turn the outside page to help page.

void **on\_checkBoxHelpPage\_clicked**(bool clicked):

Parameters: None.

Return: None.

Task: Handle the operation after the checkbox in help page clicked.

void **refreshDeviceInfo**():

Parameters: None.

Return: None.

Task: Refresh GPU information and disk information in main window and save it to database.

void **onGPUInfosReceived**(QList<GPUInfo> gpusinfo):

Parameters: GPUs’ information.

Return: None.

Task: Refresh the buffer of the GPU information and call functions to refresh the data displayed in main windows.

void **onNvMonitorInfo**(unsigned int gpucount

, unsigned int maxgputemp

, unsigned int mingputemp

, unsigned int maxfanspeed

, unsigned int minfanspeed

, unsigned int maxmemclock

, unsigned int minmemclock

, unsigned int maxgpuclock

, unsigned int mingpuclock

, unsigned int maxpowerdraw

, unsigned int minpowerdraw

, unsigned int totalpowerdraw

):  
 Parameters: detailed information of each GPU.

Return: None.

Task: Refresh the GPU information displayed in main window.

void **onHelp**():  
 Parameters: None.

Return: None.

Task: Show the help dialog if pressed the help button in the menu of the tray icon.

void **onHrChartTimer**():

Parameters: None.

Return: None.

Task: Append current hash rate to the hash rate chart.

void **onMouseHoverSlice**(QPieSlice \* slice, bool status):

Parameters: Slice the mouse hovered and its status.

Return: None.

Task: Enlarge the target slice in the temperature pie chart which mouse hovered.

void **on\_horizontalSliderPowerPercent\_valueChanged**(int value):

Parameters: The new value of the slider of the power percent in overclocking page.

Return: None.

Task: Update the settings for the GPU in the overclocking settings and change the value displayed next to the slider.

void **on\_horizontalSliderGpuOffset\_valueChanged**(int value):

Parameters: The new value of the slider of the GPU offset in overclocking page.

Return: None.

Task: Update the settings for the GPU in the overclocking settings and change the value displayed next to the slider.

void **on\_horizontalSliderMemOffset\_valueChanged**(int value):  
 Parameters: The new value of the slider of the memory offset in overclocking page.

Return: None.

Task: Update the settings for the GPU in the overclocking settings and change the value displayed next to the slider.

void **on\_horizontalSliderFanSpeed\_valueChanged**(int value):

Parameters: The new value of the slider of the fan speed in overclocking page.

Return: None.

Task: Update the settings for the GPU in the overclocking settings and change the value displayed next to the slider.

void **on\_comboBoxDevice\_activated**(int index):  
 Parameters: The new device number of the GPU for overclocking.

Return: None.

Task: Update the settings for the GPU in the overclocking settings.

void **on\_pushButtonOCPageApply\_clicked**():  
 Parameters: None.

Return: None.

Task: Overclock the GPU with given settings.

void **on\_checkBoxAutoSpeedFan\_clicked**(bool checked):  
 Parameters: Whether the check box of auto fan speed is checked or not.

Return: None.

Task: Auto control the fan speed if the check box for auto fan speed is clicked.

void **on\_spinBoxTemperature\_valueChanged**(int value):  
 Parameters: New temperature limit.

Return: None.

Task: Update the temperature settings for the GPU in the overclocking settings

void **on\_pushButtonAutoOC\_clicked**():  
 Parameters: None.

Return: None.

Task: Program take control of the overclocking with the advices settings.

Key Public Variables:

|  |  |
| --- | --- |
| QMap<QString, Coin\*> map\_coins; | Maps from string to coins. |
| QMap<QString, Core\*> map\_cores; | Maps form string to core. |
| QMap<QString, Pool\*> map\_pools; | Maps from string to pool. |

Public Types:

None.

Public Methods:

void ***setVisible***(bool visible) Q\_DECL\_OVERRIDE:

Parameters: whether main window is visible.

Return: None.

Task: Set the main window visible or not.

void **startMiner**():

Parameters: None.

Return: None.

Task: Start miner or press the start mining button.

GPUInfo **getAverage**(const std::vector<GPUInfo>& gpu\_infos):

Parameters: All GPUs’ information.

Return: Average GPU information.

Task: Calculate and return the average information of the given GPU list.

GPUInfo **getWorst**(const std::vector<GPUInfo>& gpu\_infos):

Parameters: All GPUs’ information.

Return: The largest data of each settings in the GPU list.

Task: Return largest data of each settings in the GPU list.

bool **getMinerStatus**():

Parameters: None.

Return: None.

Task: Return whether the miner is running.

void **SetUIRefresh**(bool enabled):

Parameters: Enable or disable the UI refresh function.

Return: None.

Task: Enable or disable the UI refresh function.

bool ***eventFilter***(QObject \*obj, QEvent \*event):

Parameters: The Object the event happens and the event.

Return: Whether the event happen or not.

Task: Perform some operation if some events happening on target object.

Public Slots:

None.

Signals:

None.

Static Public Members:

None.

Protected Methods:

void ***closeEvent***(QCloseEvent \*event) Q\_DECL\_OVERRIDE:

Parameters: The main window close event.

Return: None.

Task: If simply close the program by pressing the close button on the main window, the program will run in the background mode and the computer OS will prompt that the program is still running.

Static Protected Members:

None.