Tic Tac Toe

Performance Report

Presented to: Prof. Omar Nasr

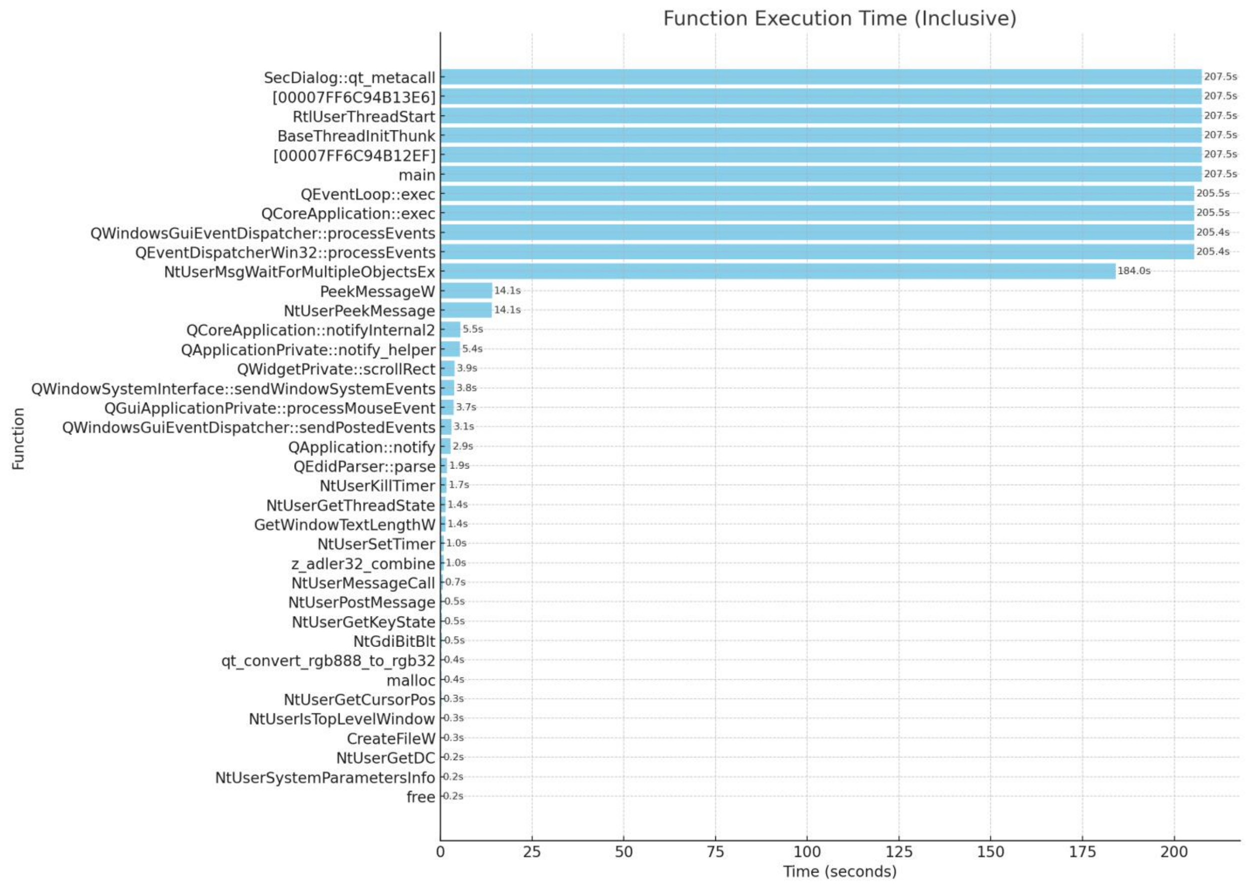
EECE27
ELECTRONICS & COMMUNICATIONS ENGINEERING

Ameer Ashraf Louly Abbass	9230242
Amir Sameh Fanous Attia	9230243
Mohamed abdullah farouk abdullah	9230787
Rola refaat bekhit	9230382
Nourhan Mohammed Fahmy	9230966

1. Testing method and evaluation

-to perform the testing we used the “very sleepy” program to monitor each task in our program and the time it consumes allowing us to evaluate the response time and the optimizations needed.

Data extracted from Very Sleepy:



In the following table rank the functions by the time they consume:

Rank	Function
1	SecDialog::qt_metacall(...)
2	QCoreApplication::exec() + Event Loop Chain
3	NtUserMsgWaitForMultipleObjectsEx
4	PeekMessageW + NtUserPeekMessage
5	QCoreApplication::notifyInternal2()
6	QApplicationPrivate::notify_helper()
7	QWidgetPrivate::scrollRect()
8	QGuiApplicationPrivate::processMouseEvent(...)
9	QWindowsGuiEventDispatcher::sendPostedEvents()
10	QApplication::notify(...)
11	QEdidParser::parse(QByteArray)

the following table explains what is the role of each function and the time it consumes:

Rank	Role	Time (%)	Time(S)
1	Handles (game window) slots	96.67%	207.53
2	dispatches all UI and system events	95.70%	205.45
3	Windows API: waits for events/messages	85.72%	184.00
4	Windows messages (mouse clicks, timers)	6.57%	14.1
5	Dispatches a Qt event to the target object	2.56%	5.50
6	Helper for delivering events to widgets	2.50%	5.37
7	Handles scrolling for a widget (probably used in your UI updates)	1.79%	3.86
8	Handles mouse movement/clicks	1.71%	3.67
9	Sends queued (posted) events	1.42%	3.05
10	Top-level event notifier for UI components	1.33%	2.86
11	Parses monitor/display capabilities	0.87%	1.86

2. Evaluation and optimization

Evaluation

This profile shows that:

The CPU is mostly idle, waiting for input (NtUserMsgWaitForMultipleObjectsEx). Most CPU time is in the main event loop, not in computation-heavy routines. Active logic (e.g., rendering, input handling) takes only a few seconds in total. So the processor usage is very low, dominated by GUI idle waiting.

Over 85% of the total execution time is spent in the event loop waiting for user input, primarily within “NtUserMsgWaitForMultipleObjectsEx”. This indicates that the application is well-behaved during idle states and not consuming unnecessary CPU.

The function SecDialog::qt_metacall(...) appears prominently in the profile, suggesting extensive use of Qt’s signal/slot mechanism, possibly for frequent or minor events.

Memory and I/O operations are lightweight; functions like malloc, free, CreateFileW, and GetWindowTextLengthW contribute insignificantly to overall time, confirming efficient resource usage.

Optimizations

-The signals and slots connections and communications is over used for frequent events

Solution: reduce the unnecessary signals from some UI elements, also we may use direct connections like “Qt::DirectConnection”

-We use frequint timers for history review which could increase the GUI thread load

Solution: we may add “next” and “previous” buttons so the user can see the game replay as he desires instead of replaying the game automatically like a video.