# Introduction

Performance for any app is crucial, however, it is extremely important that our game is responsive and efficient to ensure an enjoyable user experience and keep the player entertained. As a result, we used Visual Studio's performance profiler to identify areas of the app that could slow down our application and consume excess memory. We can then use this information in aiding the refinement of our game for future releases. It's important to note that profiling was done using debug mode and thus is not entirely representative of a release mode profile.

# CPU Usage

To measure the CPU usage of various functions, we played through an in-game day and tabulated the amount of CPU time the functions used, these results are shown below in *table 1*. The data in *table 1* was trimmed down from the complete list provided by Visual Studio, as it contained a vast amount of functions outside of our scope - GUI functions provided by Qt - therefore only the top results were searched through and our functions were extracted to produce the table.

Immediately, it is obvious that the scalable label class is consuming a lot of the CPU in determining the correct font size to use. This is something that should be adjusted for a proper release. It may mean that the scaling is not as accurate, or potentially a more efficient algorithm needs to be used at the expense of a more scalably responsive, but less computationally expensive, interface.

Other than the scalable label, the amount of CPU time our code uses is vastly inferior to that of the GUI library that we are using, mainly functions waiting for user input which creates a skewed representation. Therefore, it's clear that the majority of our code is optimised with certain aspects such as painting - drawing - elements consuming most of the CPU time, something that is also out of control as these mostly contain calls to the Qt library.

# Memory Usage

Visual Studio's snapshot functionality allowed us to take a look into the memory allocation of our program to see what is consuming the most memory, this data is represented in *table 2*. It also provided a graph of the memory usage over time, *figure 1* shows this graph over the same time span of the CPU usage - a rough in-game day.

The memory dump snapshot showed no inconsistencies and was overall a reasonable size for a game. While not immediately obvious, it's clear that some of the memory management actions that we took were working as only the visible widgets were consuming memory. For example, within the notebook widget there are multiple inner widgets acting as a menu, the hidden widgets are destroyed instead of simply hidden and as a result less memory is used.

One area of the memory that may need a further look at for future releases is the climate data as this number seems higher than expected and with multiple counts. There should, ideally, only be one active `ClimateData` for the in-game day as this struct contains multiple inner structs that can be memory consuming.

For the continuous memory usage, *figure 1* indicates that the overall memory management of the program is excellent. There is no gradual increase in memory usage, indicating no memory leaks, and there are no large spikes or other indicators that something is going wrong.

Overall, the memory usage of our program seems to be excellent and suitable even for low memory systems.

Table 1: CPU usage per function

| **Function Name** | **Total CPU [unit, %]** |
|---|---|
| main | 8974 (8.58%) |
| ScalableLabel::paintEvent | 4666 (4.46%) |
| Game::paintEvent | 875 (0.84%) |
| LevelInfoWidget::levelSelected | 548 (0.52%) |
| LevelInfoWidget::mousePressEvent | 548 (0.52%) |
| MainMenu::onNewGameButtonClicked | 548 (0.52%) |
| NewGameMenu::levelSelected | 548 (0.52%) |
| MainMenu::newGameButtonClicked | 541 (0.52%) |
| SurviveGame::onNewGame | 541 (0.52%) |
| Game::Game | 540 (0.52%) |
| NotebookWidget::paintEvent | 538 (0.51%) |
| SVGPushButton::paintEvent | 408 (0.39%) |
| SVGPushButton::SVGPushButton | 394 (0.38%) |
| Game::setupUi | 378 (0.36%) |
| ResultWidget::close | 299 (0.29%) |
| NotebookWidget::displayActionMenu | 264 (0.25%) |
| ActionMenu::ActionMenu | 216 (0.21%) |
| ActionMenu::setupUi | 216 (0.21%) |
| NotebookWidget::NotebookWidget | 201 (0.19%) |
| Game::returnToMenu | 180 (0.17%) |
| GameOver::close | 180 (0.17%) |
| SurviveGame::onReturnToMenu | 180 (0.17%) |
| MainMenu::MainMenu | 179 (0.17%) |
| Menu::Menu | 179 (0.17%) |
| NotebookWidget::loadGraphics | 155 (0.15%) |
| Menu::loadGraphics | 154 (0.15%) |
| Game::loadGraphics | 102 (0.10%) |
| Game::onResultAcknowledged | 101 (0.10%) |
| NotebookWidget::resultAcknowledged | 101 (0.10%) |

Table 2: Memory Usage

| Object | Count | Size (Bytes) |
|---|---|---|
| Standard Tree Node (Climate Data) | 13 | 1352 |
| Standard Tree Node (Widget Info) | 18 | 1296 |
| Standard Tree Node (String Pair) | 13 | 1040 |
| Climate Data | 12 | 816 |
| Standard Container Proxy | 47 | 752 |
| SVG Push Button | 7 | 672 |
| Standard Tree Node (Animal Info) | 6 | 480 |
| Standard Tree Node (Event Info) | 6 | 480 |
| String Array | 1 | 480 |
| Game | 1 | 424 |
| Standard Tree Node (Explore Info) | 5 | 400 |
| Standard Tree Node (Plant Info) | 3 | 240 |
| SVG Widget | 5 | 200 |
| QSlotObject (Game) | 6 | 192 |
| Standard Tree Node (Day Entry) | 2 | 176 |
| QSlotObject (Notebook Widget) | 5 | 160 |
| Notebook Widget | 1 | 160 |
| QSlotObject (Action Menu) | 4 | 128 |
| QSlotObject (Widget) | 4 | 128 |
| Property Animation | 7 | 112 |
| SVG Renderer | 7 | 112 |
| Graphics Opacity Effect | 7 | 112 |
| Parallel Animation Group | 7 | 112 |
| QMap Data (Widget Info) | 3 | 96 |
| Action Menu | 1 | 80 |
| Scalable Label | 1 | 40 |
| QMap Data (Animal Info) | 1 | 32 |
| QMap Data (Plant Info) | 1 | 32 |
| QMap Data (Event Info) | 1 | 32 |
| QMap Data (Explore Info) | 1 | 32 |
| QMap Data (String Pair) | 1 | 32 |
| QSlotObject (Survive Game) | 1 | 32 |
| Day | 1 | 32 |
| QMap Data (Day Entry) | 1 | 32 |
| QFunctorSlotObject (Notebook Widget) | 1 | 24 |

Figure 1: Memory usage of an in-game day