Running ThreadSanitizer (TSan)

Overview

This project utilizes ThreadSanitizer (TSan) to detect threading-related issues and ensure thread safety in the MrSnowman project. The `run_tsan.sh` script automates the process of enabling TSan, compiling the code with the necessary flags, and running the application to detect threading errors.

Enabling ThreadSanitizer

To enable ThreadSanitizer for the project, execute the `run_tsan.sh` script:

```
./run_tsan.sh
```

This script modifies the `MrSnowman.pro` file to include TSan flags, cleans the project, reconfigures it using `qmake`, rebuilds the project with TSan enabled, and then runs the application to detect threading issues.

ThreadSanitizer Process

The `run_tsan.sh` script performs the following steps:

- **Modify .pro File:**
  - The script appends the necessary TSan flags (`-fsanitize=thread`) to the `MrSnowman.pro` file to enable TSan during compilation and linking.
- **Clean and Rebuild Project:**
  - The project is cleaned, reconfigured using `qmake`, and then rebuilt using `make` with TSan flags.
- **Run Application with ThreadSanitizer:**
  - The script executes the application with TSan enabled to detect threading-related issues.

Interpreting ThreadSanitizer Results

After running the `run_tsan.sh` script, the terminal output will display any threading-related issues detected by ThreadSanitizer. An example output indicating a data race could look like this:

```
ThreadSanitizer: data race (pid=12345)
  Write of size 4 at 0x7f7fffe0e018 by thread T2:
    #0 WorkerThread(void*) my_thread.cpp:10 (my_app+0x123456789)

  Previous read of size 4 at 0x7f7fffe0e018 by thread T1:
    #0 MainThreadFunction() main.cpp:20 (my_app+0x987654321)
  Thread T2 (tid=67890, running) created at:
    #0 pthread_create my_thread.cpp:50 (my_app+0x111111111)
```

In this example:

- **data race**: Indicates a potential concurrency issue where two threads (`T1` and `T2`) are accessing shared memory (`0x7f7fffe0e018`) without proper synchronization.
- **Write of size 4 by thread T2**: Indicates a write operation by thread `T2`.
- **Previous read of size 4 by thread T1**: Indicates a read operation by thread `T1`.
- *MainThreadFunction() and WorkerThread(void)**: Function calls in the stack trace where the data race occurred.

If no issues are found, the application will run normally without any TSan error messages.

Note

It's important to note that the macOS memory allocation warnings and Qt-related messages displayed in the terminal output, such as window positions and font operations, are unrelated to ThreadSanitizer (TSan) and are provided for informational purposes only.