STM for games

Functional Specification

Author: Arkadiusz Bielecki
ID: C00139358
Date: 12.12.2013 Project Supervisor: Joseph Kehoe

Table of Contents

1. Introduction 3
2. Functionalities 3
3. Potential Users 5
4. Metrics 6
5. Similar projects 6
6. FURPS+ 7

1. Introduction

The main purpose of my project is to show how Software Transactional Memory (STM) (a concurrency control mechanism) will affect the performance in gaming. I will implement Simple and Fast Multimedia Library on a board game, using a standard synchronization, then using various STM algorithms, benchmark them, and compare whether the performance have increased or not. The performance will be measured under 1-, 2-, and 4-core processor configurations and measurement will be based on the screen refreshing rate (in frames per second - more fps means that the configuration runs faster and smoother).

2. Functionalities

The main function of application will be based on a simple board game which contains enemies and obstacles, written in C++ using Simple and Fast Multimedia Library. Another measurement will be based on a similar game, but containing a bigger playing "board", with more enemies and obstacles. The performance will be measured on small, medium and big playing "board" using various algorithms on each.

All of the different configurations will consist of different configurations described below:

Algorithms used:

- **Swarming** - it was developed from observing the animal world. Its based on self-organisation, interactions, feedback and fluctuations.

- **Flocking** - has been also taken from the animals interactions.
  Basic models of flocking behavior are controlled by three simple rules:
  1. Separation - avoid crowding neighbors (short range repulsion)
  2. Alignment - steer towards average heading of neighbors

3. Cohesion - steer towards average position of neighbors (long range attraction)

- **A*** - uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way.

<u>Run on:</u>

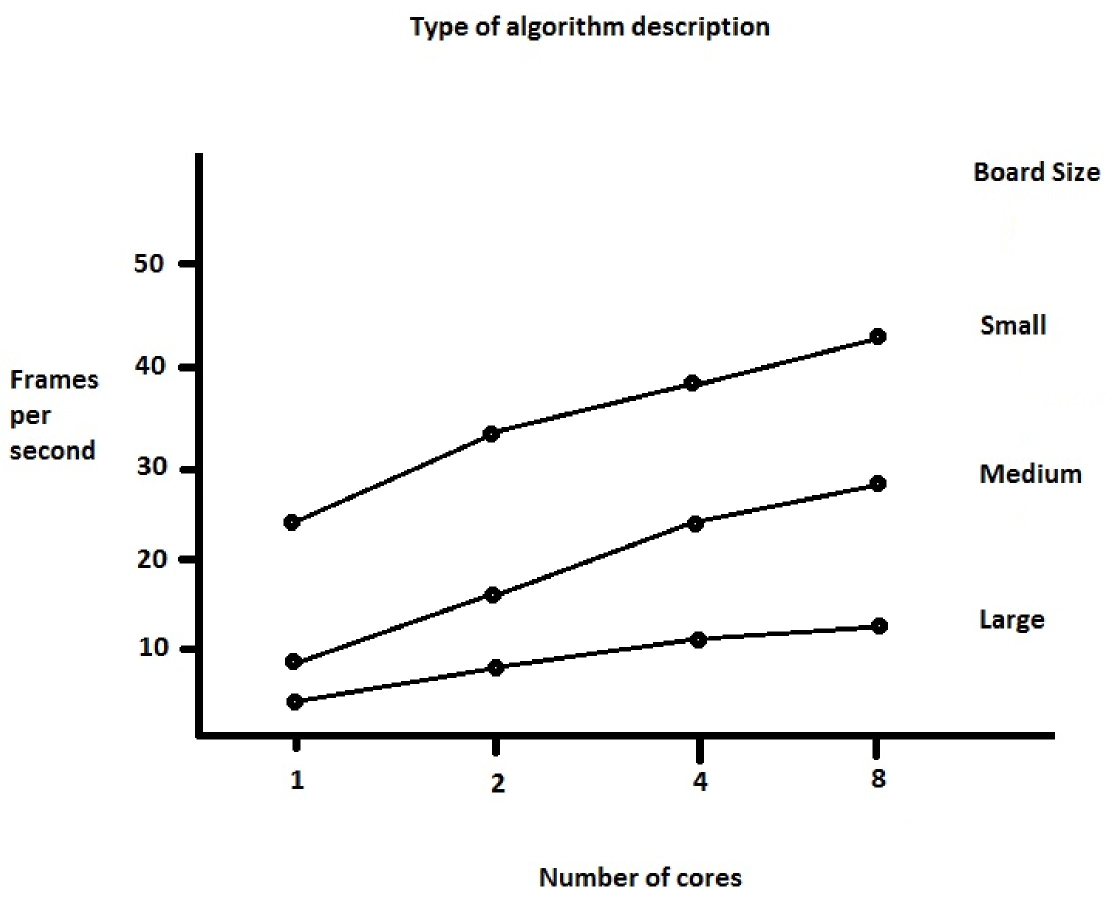
- 1 processor core
- 2 processor cores simultaneously
- 4 processor cores simultaneously
- 8 processor cores simultaneously

<u>Using ”board” size:</u>

- **Small** - board size of 640×480 pixels used to measure the output of performance.
- **Medium** - board size of 1280×800 pixels used (more calculations needed).
- **Large** - board size of 1600x1200 pixels used (even more calculations needed).

Frames Per Second will be measured under every configuration and shown on the appropriate diagram for comparison.

Below I have shown how an example output graph will look like:



3. Potential Users

Software Transactional Memory can have a variety of uses. My project is concerned around computer games, but STM can be used in broad kinds of software. It can be used in any program or game that runs parallel processes, and currently almost all of them meet that criteria. It’s used by programmers that code the application, in order to find out if it will be worth to implement it in their project.

4. Metrics

In order to gauge my project as successful I need to fulfil the **mandatory** requirements specified below:

- Integration of STM library with SFML
- Benchmarking of STM on sample game algorithms

To get a higher mark for my project I must implement some or all of the following specs:

Discretionary requirements

- Benchmarking and comparison of different STM strategies
- Integration of game algorithms into a sample game to use as a performance test bed

Exceptional requirements

- Publishable results of the effects of different STM algorithms on games

5. Similar projects

There was a number of similar projects done in the past which implemented Software Transactional Memory measurements. Some of them were concerned about games, others on different software.

My project will be very useful and quite simple at the same time. I’m gonna test STM under many various conditions, using a range of algorithms. The results will be shown on a chart that is easy to read from. That will be the main functions that differ from similar projects already out there. After looking at the graphs that my project produced, any software developer will be able to choose the best algorithm for his application, meeting their configuration requirements.

6. FURPS+

1. Introduction

The Supplementary Specifications capture the system requirements that are not captured in the use cases or other documents. Such requirements include:

- functionality
- usability
- reliability
- performance
- supportability
- and +

2. Functionality

System Error Logging

All errors should be logged to persistent storage.

3. Usability

A user should be able to see the appropriate output graph after <10 seconds, 90% of the time. Minimum target platform is Windows XP. Minimum screen size 800x480.

4. Reliability

Out of 100 Activities launched how many should fail and have to be restarted by the user?
Mean Time Between Failures shall exceed 300 hours.

5. Performance

In order for the user experience to be as smooth as possible, the graph outputs should happen in less than 10 seconds.
The app should respond in 5 seconds at most after the user selects appropriate data.
The app should not take too much resources (CPU, RAM).

6. Supportability

Automatic error logging and reporting back to the developer.
Usage reporting.

7. +

GUI

User Interface and graphs should be clear to read from.