Dual State Framework

Functional Specification

Yu Chen - C00151352

2015/04/06 23:55:58

Contents

Chapter 1

Introduction

1.1 Purpose

The main purpose of my project is to create a C++ framework for parallel computing. Parallel computing is the science and art of programming computers that can do more than one operation at once during the same cycle, simultaneously and concurrently. It performs often via having more than one processor.
Next, A benchmark program is required for profiling this framework. The program will be designed as a GU↩
I application which shows the performance in the following list of deferences.

- Number of threads
- Number of objects

1.2 Potential Users

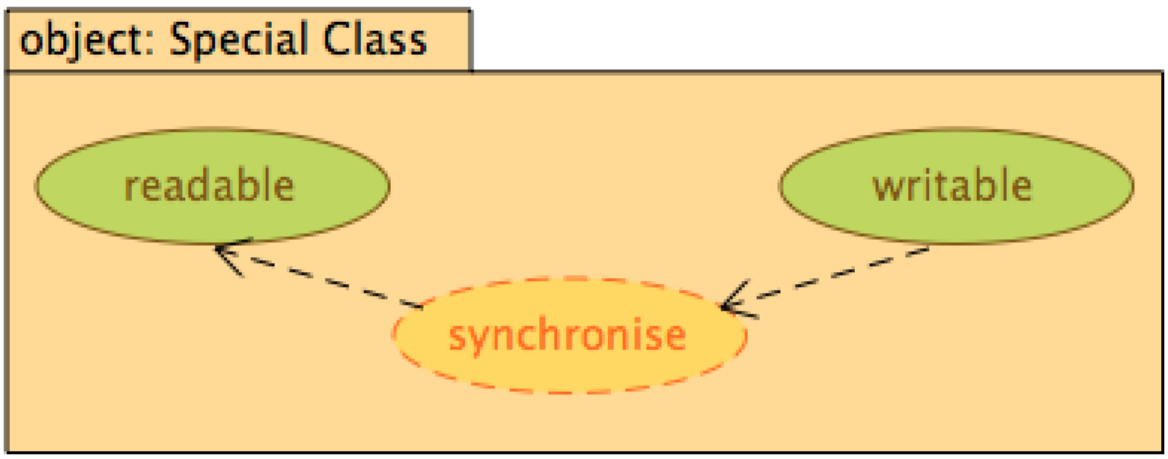
Dual State Framework can have a variety of uses. Initially the project is concerned around computer games, but it can be used in broad kinds of software. The framework can be used in any program or game that runs parallel processes. Currently almost all of them meet that criteria.

Chapter 2

Functionalities

2.1 Dual State

A special class should be designed for the framework. Each object of the class has two states for read operation and write operation. The purpose of the class is for thread security. In each cycle, the state of read operation should not be changed. Users can only change the state of write operation. After this cycle, these two states will be synchronised.



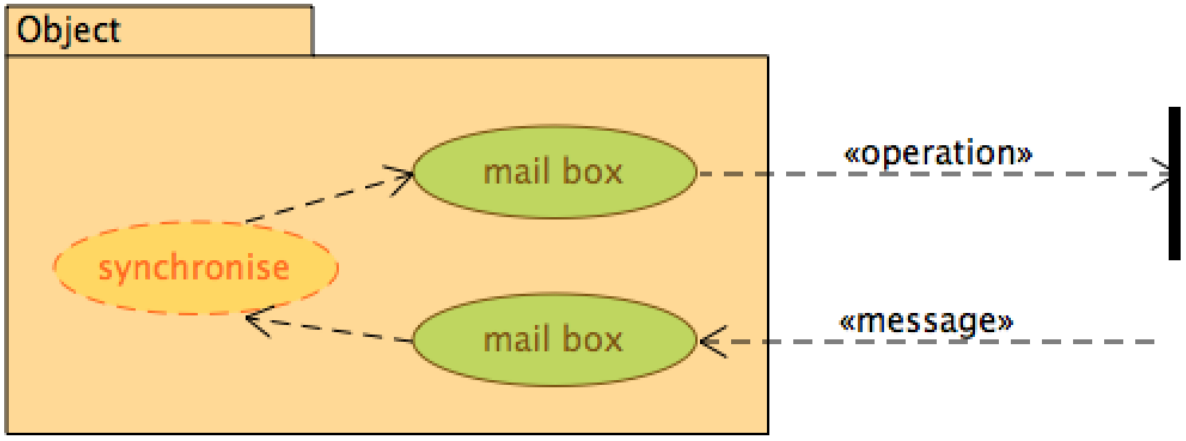
2.2 Message Delivery

Objects in the framework should be able to send messages to other objects. Each message contains: where this message sent from

- operation
- arguments for the operation

2.3 Mailbox

The mailbox stores messages that object received. It should be designed as Dual State. Therefore, each object will have two copies of mailbox. One is for performing the operation, and the other one is for receiving messages.


Object
mail box
synchronise
mail box
«operation»
«message»

Chapter 3

Benchmark

3.1 Graphics User Interface

The benchmark program should be designed as a GUI application. Also, it should allow users to configure detail settings such as:

- Max number of threads
- Number of objects
- Methods
- Duration

3.2 Methods

The program should has more than one methods to profile the framework. The following algorithm is recommended.

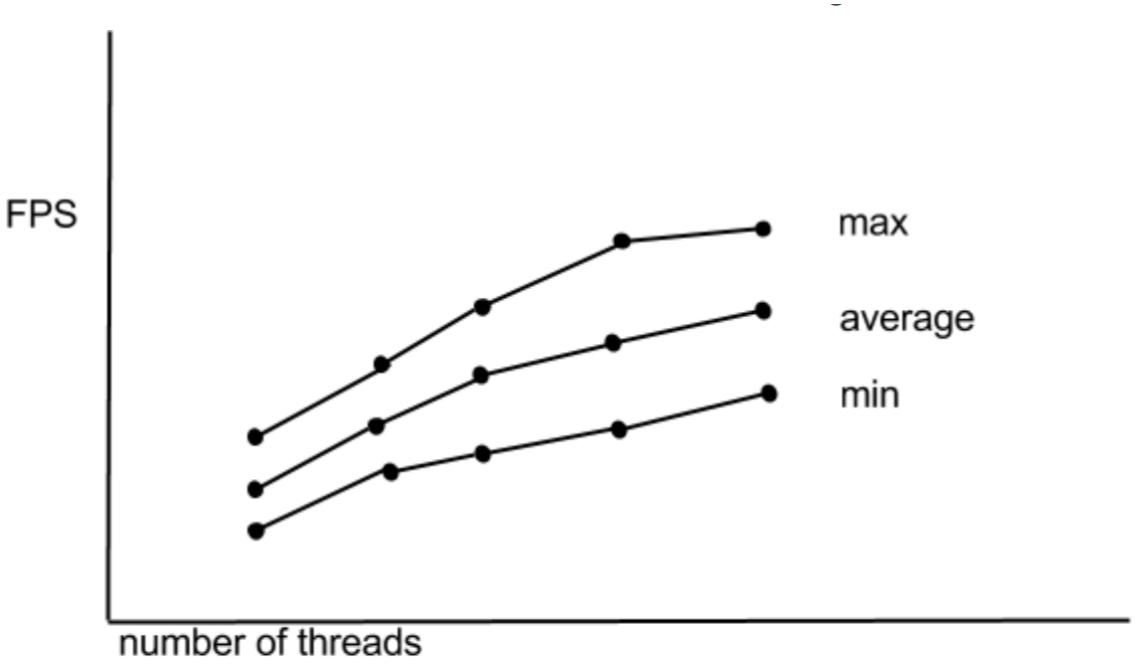
- Random objects
  - Objects appear in random positions
  - Positions change every frame
- Collision
  - Objects locate to random position
  - Each object has a random velocity
  - After collision, objects get new velocities
- Flocking boids
  - Separation - avoid crowding neighbors
  - Alignment - steer towards average heading of neighbors
  - Cohesion - steer towards average position of neighbors

3.3 Outputs

The output will be a graph which shows all information of the benchmark.

- x axis representing number of threads

- y axis representing frames per second(FPS)
- values with the maximum, the minimum, and the average.



Chapter 4

FURPS

4.1 Functionality

The framework will run on multiple platforms. Source code will be able to be compiled on varies of C++ compilers. In addition, installers for different operating system will be created. Popular IDE and OS will be supported.

- Mac OS X - Xcode
  - .dylib dynamic library
  - .framework bundled library
- Windows - Visual Studio
  - .lib compiling time dynamic library
  - .dll runtime dynamic library
- Linux - Makefile
  - .so dynamic library

4.2 Usability

User does not need to know the code behind the scene because that is not important. User who has experienced on C++ should be easy to understand the framework. An API document should be created. The document should contain: Installation

- Namespace Documentation
- Class Documentation
- Simple examples

4.3 Reliability

The framework should run correctly after installation. Memory leaks should not occur at runtime. Deadlock or livelock is not allowed. A deadlock is where a system locks up because two or more processed are waiting for each other to finish. Livelock is similar to deadlock, but except that the states of the processes involved in the livelock constantly change with regard to one another.
To know more about memory leak, deadlock and livelock requires a high level of programming skill that scriptures cannot be assumed to have. The model allows the user do not need to worry about these issues.

4.4 Performance

The best performance of this framework should be based on the running machine. For example, the CPU of my laptop is intel i5, which has two cores and executes four threads. The best performance of my laptop is when I use this framework with four threads.
The framework should not consume too much resources(CPU, RAM).

4.5 Supportability

Ensure that the product can be localized later for different languages and units later on. The maintenance of the system shall be done as per the maintenance contract. Update will be display when it is available.