7 User Manual

7.1 Introduction to the application

The application provides an animation of the classical producer-consumer problem in computer science.

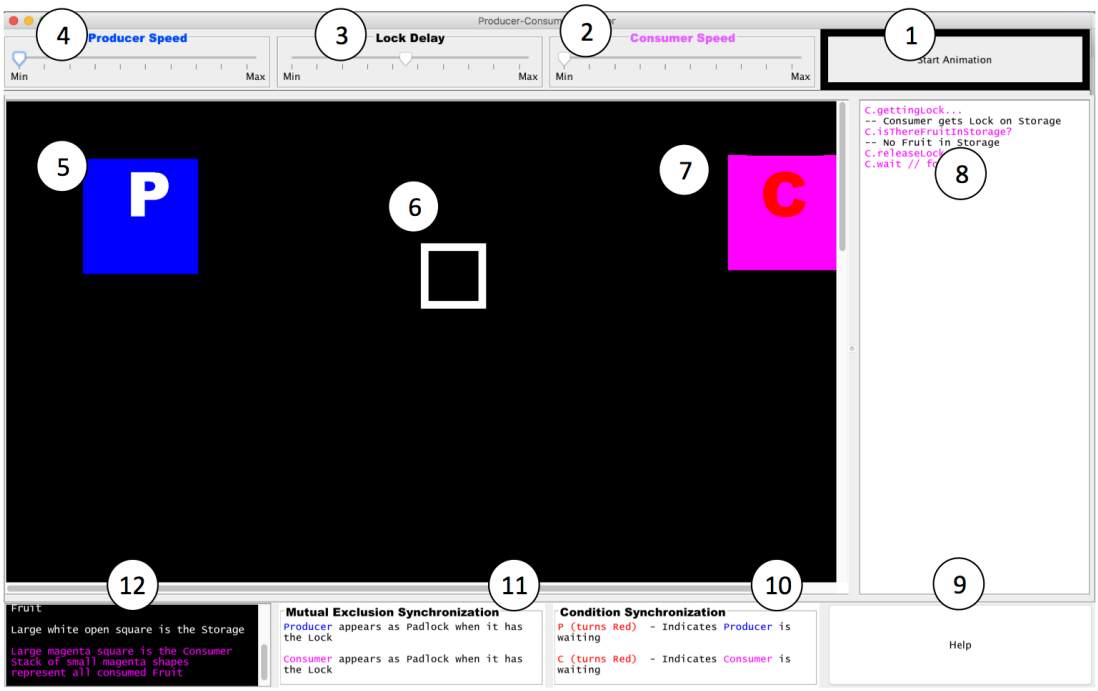
The producer-consumer problem is a classic example of a multiple thread synchronisation problem. The problem describes two threads, the Producer and the Consumer, who share a single slot in an object called Storage. The Producer's job is to create a Fruit object and store it in Storage's single slot. At the same time the Consumer is removing the Fruit object from Storage's single slot. The problem is to make sure the Producer doesn't try to add a Fruit to a full slot and the Consumer doesn't try to remove a Fruit from an empty slot.

The animation will illustrate two forms of synchronisation used in solving this version of the producer-consumer problem. The two forms of synchronisation are **mutual exclusion synchronisation** and **condition synchronisation**.

Mutual exclusion is the guarantee that access to a single shared resource (the single slot in the Storage object) is achieved on mutually exclusive basis. To ensure the Producer and Consumer can only access the Storage object on exclusive basis, access was controlled by a single lock. The lock is declared by synchronizing on the Storage object. The Producer and Consumer must acquire the lock before being able to change the state of the Storage object.

Condition synchronisation is a higher level of control which occurs after a thread has been granted exclusive access. It forces the threads to wait for a condition to become true before being allowed to change the state of the Storage (i.e. Producer inserting a Fruit, Consumer removing a Fruit). This is achieved by the Producer and Consumer threads communicating with each other using the methods `wait()` and `notifyALL()`. If the condition is false then the thread executes `wait()`, suspends its sequence of execution and gives up the lock. If the condition is true then a thread can continue its sequence of execution, change the state of the Storage and signal to the other using `notifyALL()`.

7.2 How to use this application



1) Press the "Start Animation" button to start the simulation of the producer-consumer problem.

2) To increase the speed at which the Consumer removes Fruit from Storage move the slider towards the MAX.
   To decrease the speed at which the Consumer removes Fruit from the Storage move the slider towards the MIN.

3) To increase the amount of time the Producer and Consumer hold the lock on Storage move the slider towards the MAX.
   To decrease the amount of time the Producer and Consumer hold the lock on Storage move the slider towards the MIN.

4) To increase the speed at which the Producer produces Fruit move the slider towards the MAX.
   To decrease the speed at which the Producer removes Fruit move the slider towards the MIN.

5) This icon illustrates the Producer which produces pieces of Fruit.

6) This icon represents the Storage which can hold a single piece of Fruit.

7) This icon illustrates the Consumer which consumes Fruit.

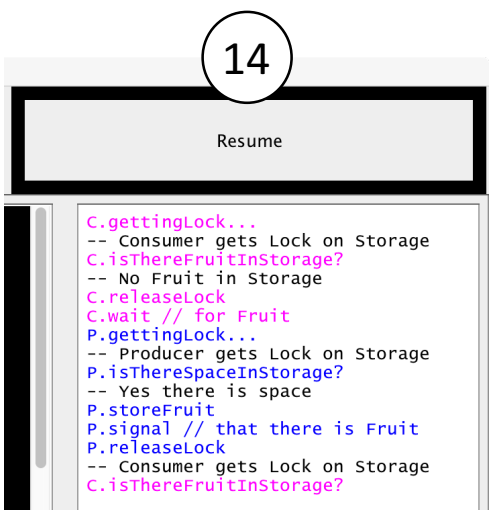
8) This panel provides the pseudocode.

9) Pressing the "Help" button provides a full description of the producer-consumer problem and how the animation illustrates the steps in this process.

10) This panel explains the colour coding indicating the waiting associated with condition synchronisation.

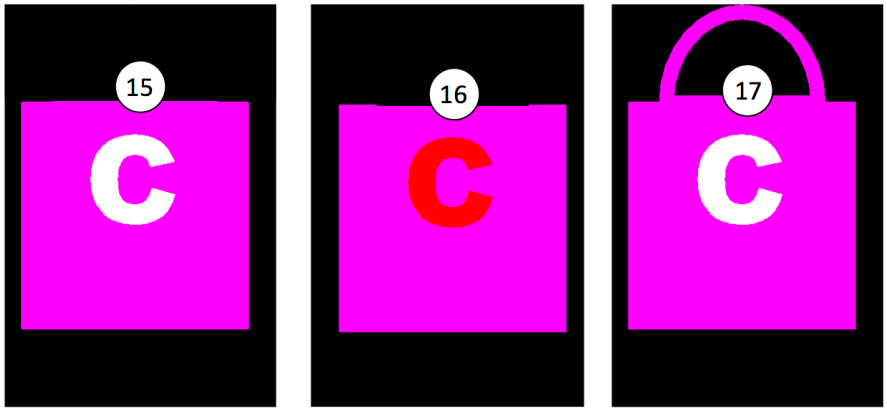
11) This panel explains the appearance of the lock associated with exclusion synchronisation.

12) This panel explains the colour coding for the animation.



13) Press the "Pause Button" to pause the Animation.

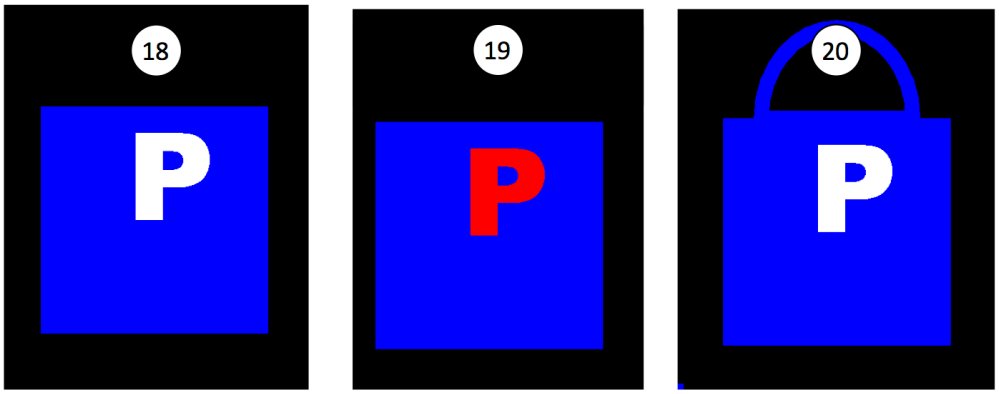
14) The Press the "Resume Button" to resume the Animation from the place it was previously Paused.



15) The Consumer icon.

16) The Consumer icon in red to indicate waiting for a signal that there is Fruit in the Storage to be removed.

17) The Consumer icon in "Lock" mode to indicate that it has acquired the Lock.



18) The Producer icon.

19) The Producer icon in red to indicate waiting for a signal that there is space in the Storage.

20) The Producer icon in "Lock" mode to indicate that it has acquired the Lock.

7.3 Hints and Tips

The two forms of synchronisation are animated in the application:

Below are recommended settings for the user to best visualise **Mutual Exclusion Synchronisation** (represented by a Lock)

Producer Speed = **MIN**
Lock Delay = **MAX**
Consumer Speed = **MIN**

These settings offer the best visualisation of Producer and Consumer acquiring the Lock. The appearance of the Producer or Consumer as a Lock represents the current thread with exclusive access.

Below are recommended settings for the user to visualise **Condition Synchronisation** (represented by P and C turning red whilst waiting for a condition to be satisfied).

Producer waiting:

Producer Speed = **MAX**
Lock Delay = **MIN**
Consumer Speed = **MIN**

These settings offer the best visualisation of the Producer waiting* for a signal** from the Consumer that there is space available in Storage.

Consumer waiting:

Producer Speed = **MIN**
Lock Delay = **MIN**
Consumer Speed = **MAX**

These settings offer the best visualisation of the Consumer waiting* for a signal** from the Producer that there is Fruit available in Storage.

*By calling `wait()`
**By calling `notifyAll()`