

# Introduction to Barrier Synchronization

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**



**Institute for Software  
Integrated Systems  
Vanderbilt University  
Nashville, Tennessee, USA**



# Learning Objectives in this Part of the Lesson

- Understand what barrier synchronization is & know three different ways of using barrier synchronizers





# Learning Objectives in this Part of the Lesson

- Understand what barrier synchronization is & know three different ways of using barrier synchronizers
- Note a human known use of barrier synchronization



---

# Overview of Barrier Synchronization

# Overview of Barrier Synchronization

---

- Earlier discussions of Java synchronizers have largely focused on classes that affect the behavior of individual threads



---

See earlier lesson on "*Types of Java Synchronizer Capabilities*"

# Overview of Barrier Synchronization

---

- Earlier discussions of Java synchronizers have largely focused on classes that affect the behavior of individual threads, e.g.
  - Atomic operations are actions that happen effectively all at once or not at all



---

See earlier lessons on "*Java Atomic Operations & Classes*"



# Overview of Barrier Synchronization

- Earlier discussions of Java synchronizers have largely focused on classes that affect the behavior of individual threads, e.g.
  - Atomic operations are actions that happen effectively all at once or not at all
  - Mutual exclusion synchronizers allow concurrent access & updates to shared mutable data within critical sections



See earlier lessons on "*Java ReentrantLock*", "*Java Semaphore*", "*Java ReentrantReadWriteLock*", "*Java StampedLock*", & "*Java Monitor Objects*"

# Overview of Barrier Synchronization

---

- Earlier discussions of Java synchronizers have largely focused on classes that affect the behavior of individual threads, e.g.
  - Atomic operations are actions that happen effectively all at once or not at all
  - Mutual exclusion synchronizers allow concurrent access & updates to shared mutable data within critical sections
  - Coordination synchronizers ensure that computations run properly
    - e.g., in the right order, at the right time, under the right conditions, etc.



---

See earlier lessons on "*Java ConditionObject*" & "*Java Monitor Objects*"



# Overview of Barrier Synchronization

- In contrast, a barrier is a synchronizer that ensures thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier



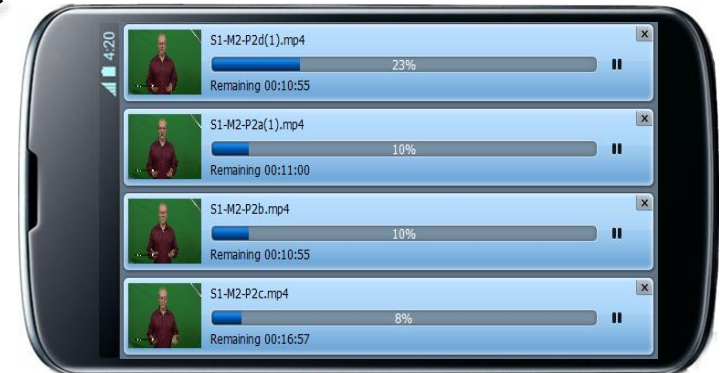
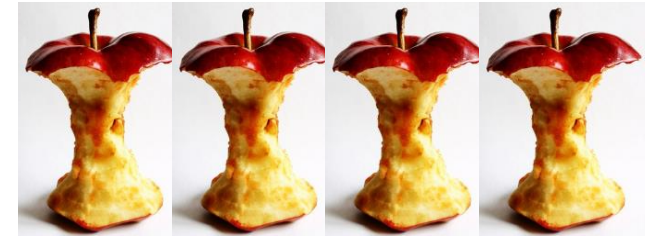
See [en.wikipedia.org/wiki/Barrier\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Barrier_(computer_science))

# Overview of Barrier Synchronization

- Barriers can be used in three ways



*We'll use a video rendering engine as a running example in this part of the lesson*



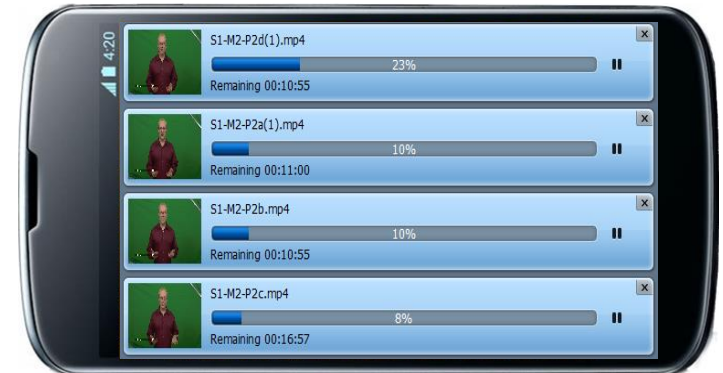


# Overview of Barrier Synchronization

- Barriers can be used in three ways

## A. Entry barrier

- e.g., keep concurrent computations from running until object(s) are fully initialized





# Overview of Barrier Synchronization

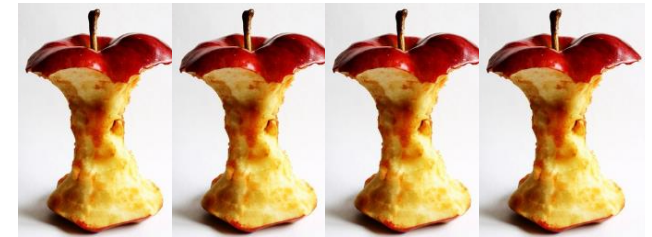
- Barriers can be used in three ways

## A. Entry barrier

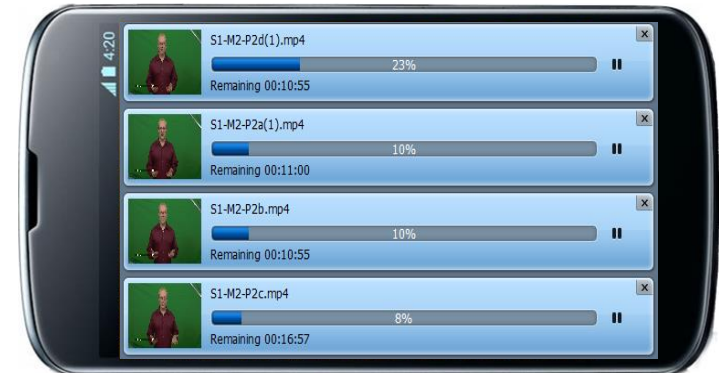
- e.g., keep concurrent computations from running until object(s) are fully initialized

Worker Threads → { → { → { → {

→ {  
Main Thread



*Main thread spawns some # of worker threads & then performs some time-consuming initialization of data structures*

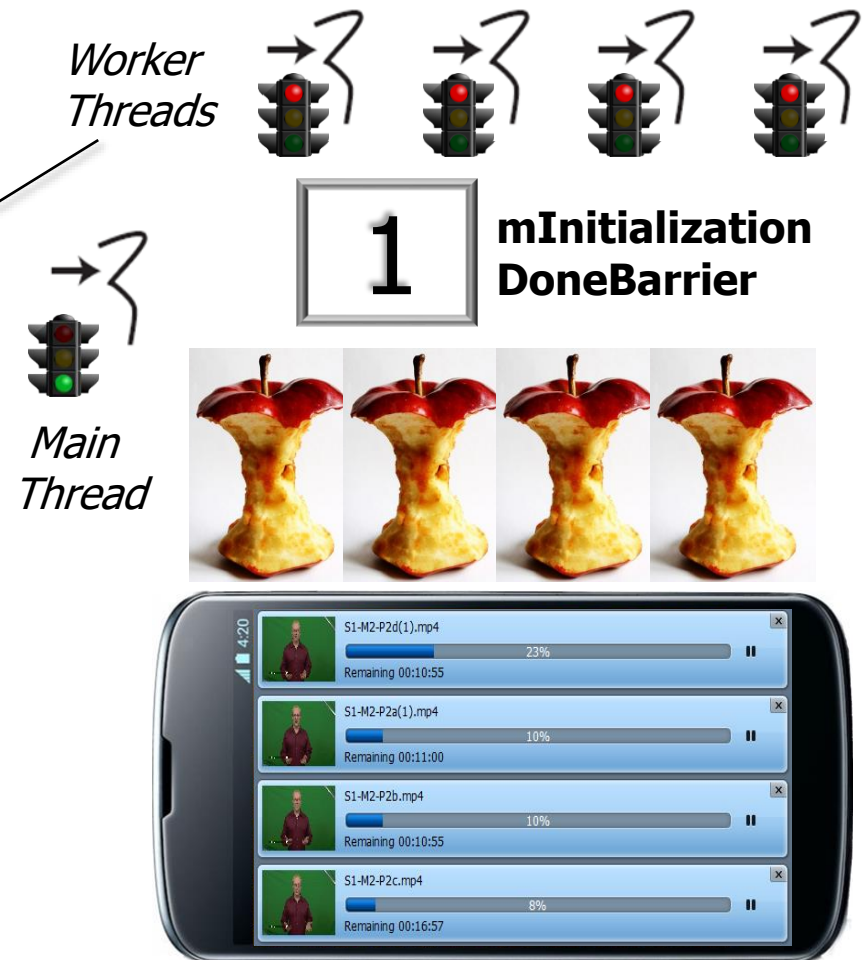


# Overview of Barrier Synchronization

- Barriers can be used in three ways

## A. Entry barrier

- e.g., keep concurrent computations from running until object(s) are fully initialized

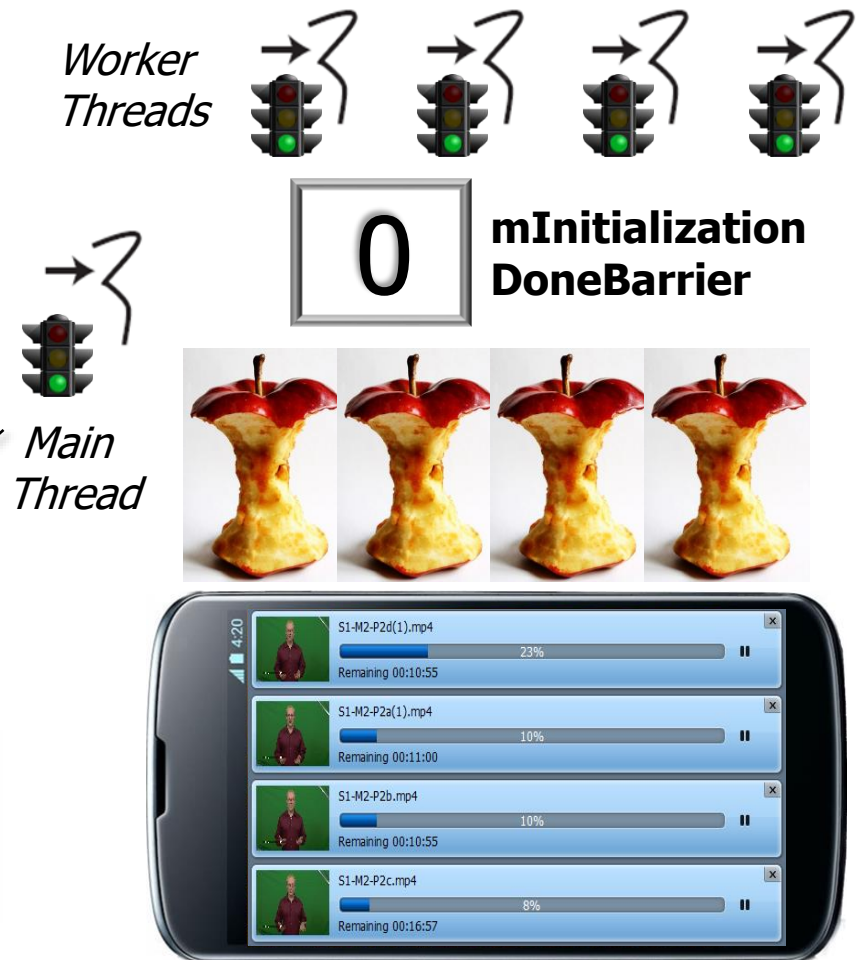


# Overview of Barrier Synchronization

- Barriers can be used in three ways

## A. Entry barrier

- e.g., keep concurrent computations from running until object(s) are fully initialized





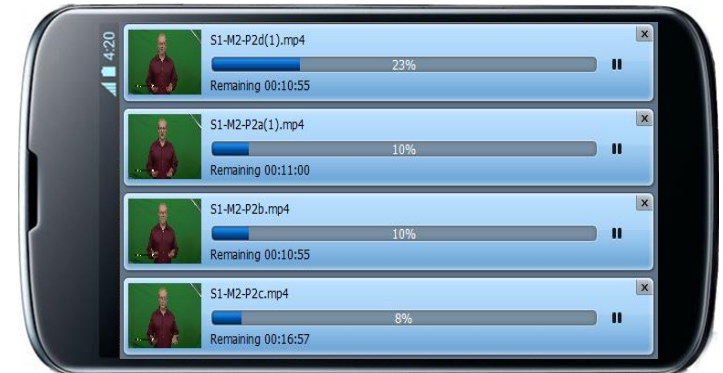
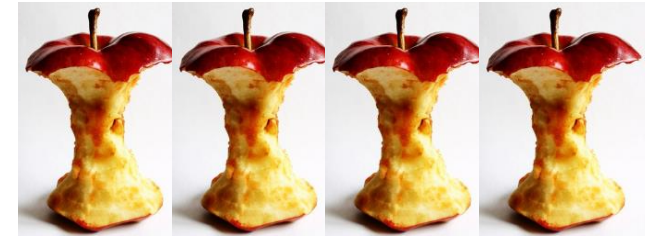
# Overview of Barrier Synchronization

- Barriers can be used in three ways

## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing



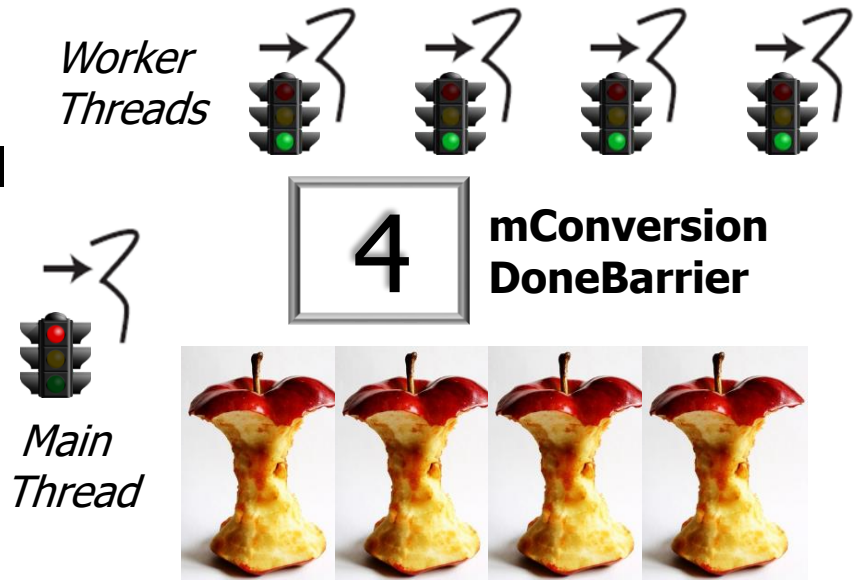
# Overview of Barrier Synchronization

- Barriers can be used in three ways

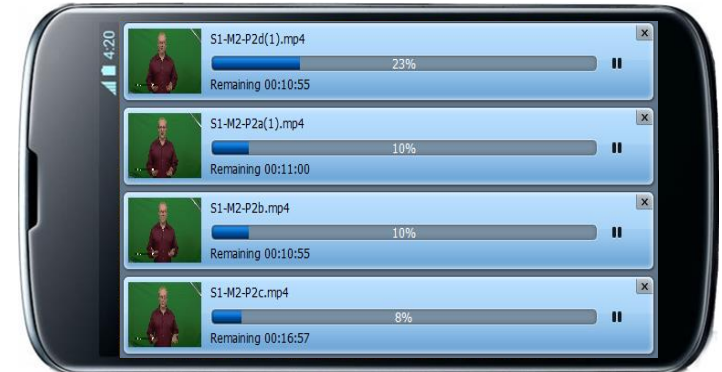
## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing



*The main thread waits on an exit barrier for all worker threads to finish*



# Overview of Barrier Synchronization

- Barriers can be used in three ways

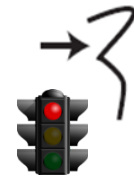
## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing

*Barrier count decrements when thread's done*

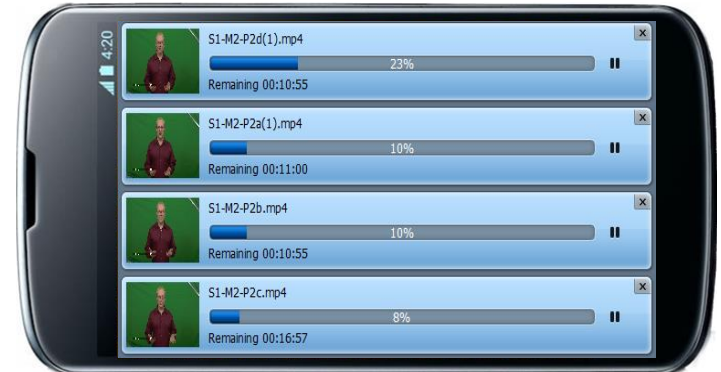
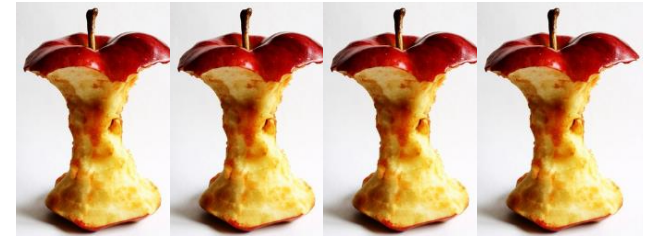
Worker  
Threads



Main  
Thread



mConversion  
DoneBarrier





# Overview of Barrier Synchronization

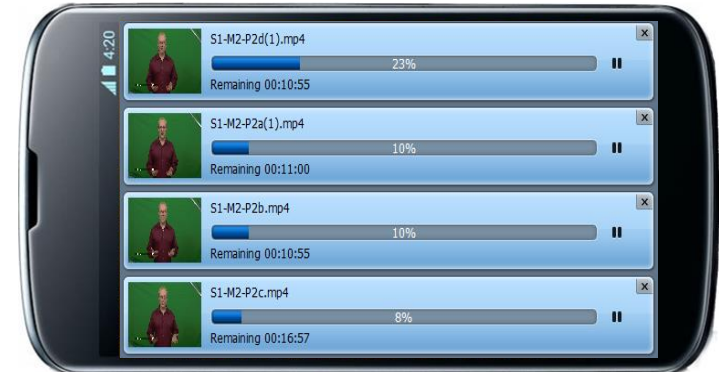
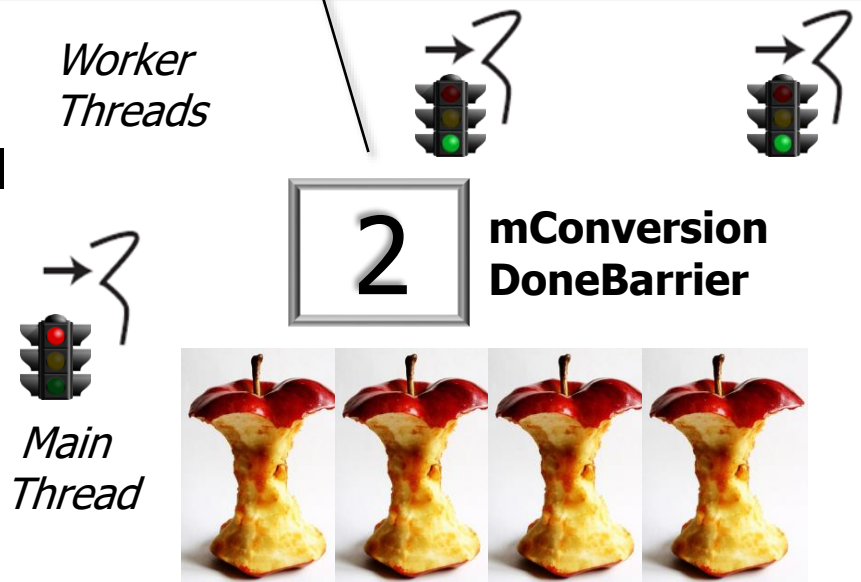
- Barriers can be used in three ways

## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing

*Barrier count decrements when thread's done*



# Overview of Barrier Synchronization

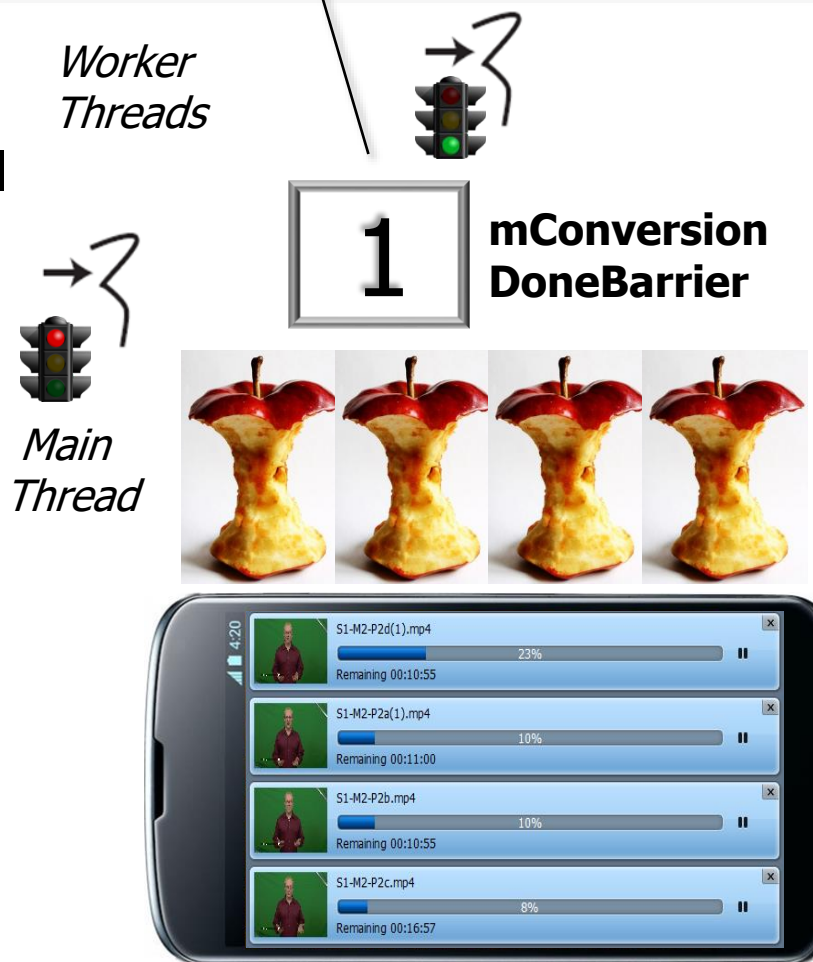
- Barriers can be used in three ways

## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing

*Barrier count decrements when thread's done*



# Overview of Barrier Synchronization

- Barriers can be used in three ways

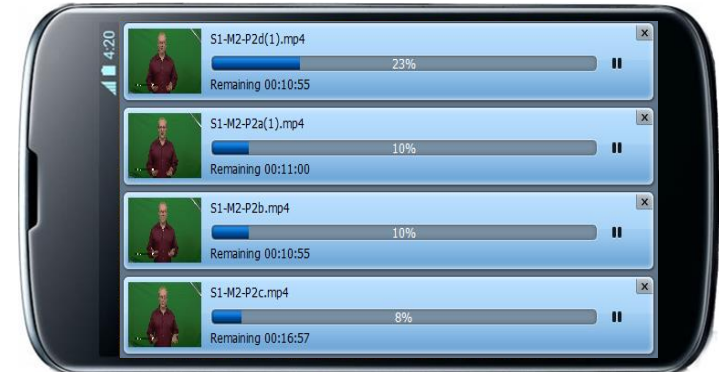
## A. Entry barrier

## B. Exit barrier

- e.g., don't let a thread continue until a group of concurrent threads have finished their processing



**mConversion  
DoneBarrier**



*When the exit barrier count = 0  
the main thread can now continue*

# Overview of Barrier Synchronization

- Barriers can be used in three ways

**A. Entry barrier**

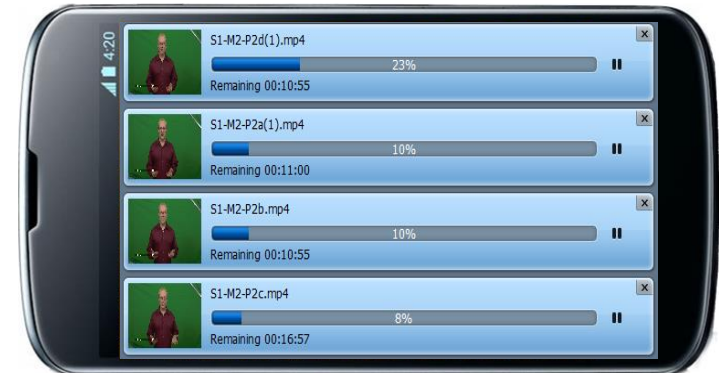
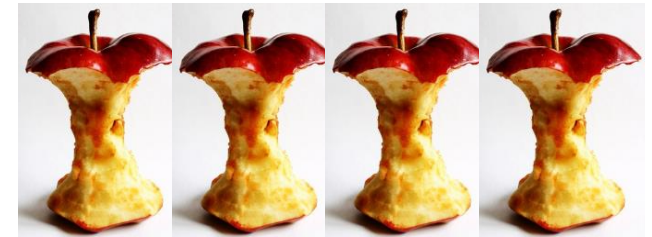
**B. Exit barrier**

**C. Cyclic barrier**

- e.g., a group of threads all wait for each other to reach a certain point before advancing to the next cycle



**mCyclic Barrier**





# Overview of Barrier Synchronization

- Barriers can be used in three ways

A. Entry barrier

B. Exit barrier

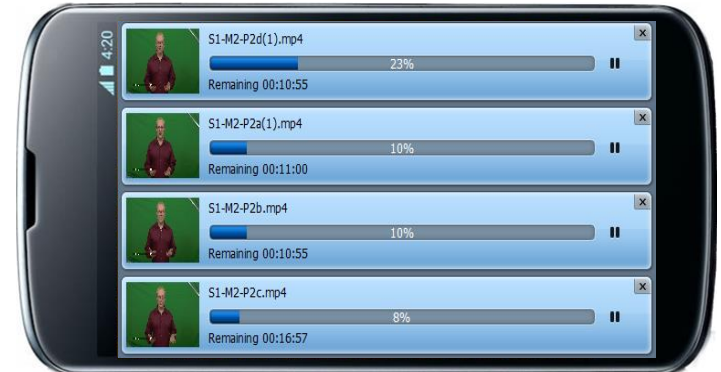
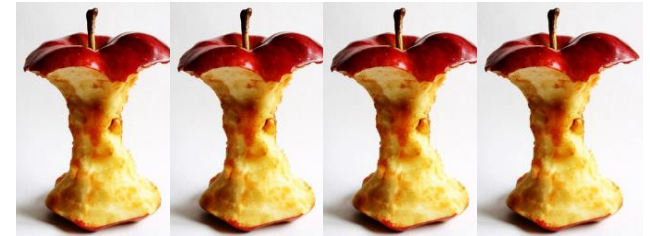
C. Cyclic barrier

- e.g., a group of threads all wait for each other to reach a certain point before advancing to the next cycle

*A fixed- or variable-size pool of threads can run concurrently*



**mCyclic Barrier**



# Overview of Barrier Synchronization

- Barriers can be used in three ways

A. Entry barrier

B. Exit barrier

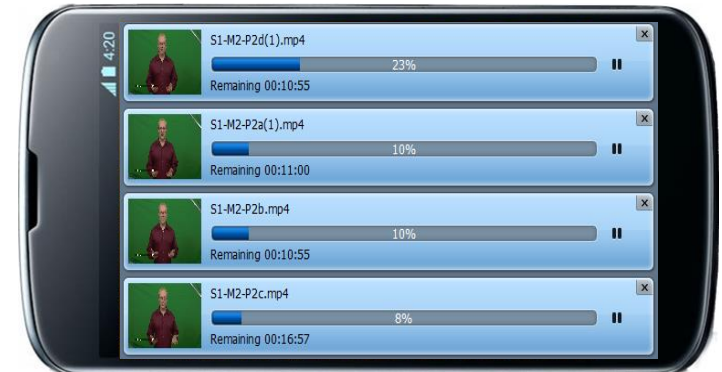
C. Cyclic barrier

- e.g., a group of threads all wait for each other to reach a certain point before advancing to the next cycle

*At the end of each cycle a decision is made about whether to continue or not*



**mCyclic Barrier**



---

# Human Known Uses of Barrier Synchronization

# Human Known Uses of Barrier Synchronization

- A human known use is protocol used by a museum tour guide



See [en.wikipedia.org/wiki/Tour\\_guide](https://en.wikipedia.org/wiki/Tour_guide)



# Human Known Uses of Barrier Synchronization

- A human known use is protocol used by a museum tour guide

## A. Entry barrier

- Tourists wait outside museum until it opens or until a tour is schedule to begin



# Human Known Uses of Barrier Synchronization

- A human known use is protocol used by a museum tour guide
  - A. Entry barrier**
  - B. Exit barrier**
    - The museum closes only after last group of tourists leave



# Human Known Uses of Barrier Synchronization

- A human known use is protocol used by a museum tour guide
  - A. Entry barrier**
  - B. Exit barrier**
  - C. Cyclic barrier**
    - Tour guide waits for all the tourists to finish exploring a room before continuing the tour in next room



Cyclic barriers can be used either as entry or exit barriers



# Human Known Uses of Barrier Synchronization

- A human known use is protocol used by a museum tour guide

**A. Entry barrier**

**B. Exit barrier**

**C. Cyclic barrier**



Barriers can be used for both fixed- & variable-sized number of tourists



---

# End of Introduction to Barrier Synchronization