Synchronization
&
Game Loop Design

Code Examination – Thread run()

```
public void run()
{
  Canvas canvas = null;
  while (mbIsRunning)
  {
    canvas = mSurfaceHolder.lockCanvas();
    if (canvas != null)
    {
        mGraphicsSurfaceView.onDraw(canvas);
        mSurfaceHolder.unlockCanvasAndPost(canvas);
    }
  }
}
```

Back to SurfaceView

- Provides a dedicated surface for a secondary thread to render screen content
- All SurfaceView and SurfaceHolder.Callback methods are called from the thread running the SurfaceView's window (typically the main application thread)

What potential thread problems can exist?

Synchronization

- Every Java object (including every class loaded) has an associated lock
- synchronized block
  - compiler adds instructions to acquire lock before executing code
  - compiler adds instructions to release lock after executing code
- thread owns the lock

More Synchronization

If thread A and thread B both have access to a Counter object and thread A owns the lock, thread B must wait for thread A to release the lock. Thus, simultaneous calls to increment and decrement behave correctly.

```
public class Counter
{
  private int count = 0;
  public void increment ()
  {synchronized (this){++count;}}
  public void decrement ()
  {synchronized (this) {--count;}}
}
```

Questions to think about

1. What is the purpose of `c = _surfaceHolder.lockCanvas(null);`
2. What is the purpose of synchronized?
3. Where do we have to use synchronized?
4. What threads exist and what are they doing?

Game Loop Design

- Games consist of:
  - getting user input
  - updating the game state (physics)
  - game AI
  - music/sound effects
  - game display

```
while (bIsRunning)  // main game loop
{
  updateGame ();
  drawGame ();
}
```

Vector versus ArrayList

- Vector
  - Synchronized
  - Thread safe
- ArrayList
  - Unsynchronized
  - Not thread safe
- Synchronization incurs a performance hit so chose wisely

Vector versus ArrayList

- Both Vector and ArrayList use an array for their contents
- A Vector defaults to doubling in size if the array is full
- An ArrayList defaults to increasing by 50% if the array is full

Code Examination – SurfaceView onTouch ()

```
public boolean onTouchEvent (MotionEvent event)
{
   return super.onTouchEvent ();
}
```

Problem #1: Add the method onTouchEvent to the SurfaceView such that when the BallAnimation application is run, the screen is black. When the user clicks anywhere on the screen, a blue ball will be centered at the tip of the cursor on the screen.

Synchronization

- Problem #2 – Allow the user the ability to place an unknown number of blue balls on the screen without any animation and without any synchronization.
- Method1: Use an ArrayList. Can you crash your program?
- Method2: Use a Vector. Can you crash your program?

Synchronization

- Problem #3 – Allow the user the ability to place an unknown number of blue balls on the screen with animation where the balls bounce off of the sides of the display.
- Use an ArrayList with the proper synchronization
- Create a method animate in GraphicsSurfaceView called from the GraphicsSurfaceViewThread class