

# Greenthreading in Flex

Presented by Huyen Tue Dao

# About me

- ✱ Name  $\approx$  “Hwin Tweh Dow”
- ✱ Fell in love with programming in a C++ class in high school.
- ✱ Flex developer since 2006.
- ✱ numTimesSpeaking = 3.
- ✱ Level 30 Infiltrator in ME2.

“I SHOULD GO...”



I AM THE STIG...OKAY, NOT REALLY BUT I DID PASS MY DRIVER'S TEST ON THE FIRST GO

# Outline

- \* The Problem
- \* Some Solutions
- \* Greenthreading
- \* Comparison
- \* Credit Where Credit Is Due
- \* Conclusion
- \* Questions

# The Problem

- \* Flash Player well suited to certain tasks: animation.
- \* Seems to be able to do several things at once: concurrency.
- \* However, it can get hung up if one of these things requires lots of processing.
- \* Does not take complex code to hang it up either.
- \* Let me show you...

# The Problem

- \* The problem revolves around Flash platform's event-driven, frame-base architecture.
- \* Flash movies/SWFs: content divided into frames played according to a timeline.

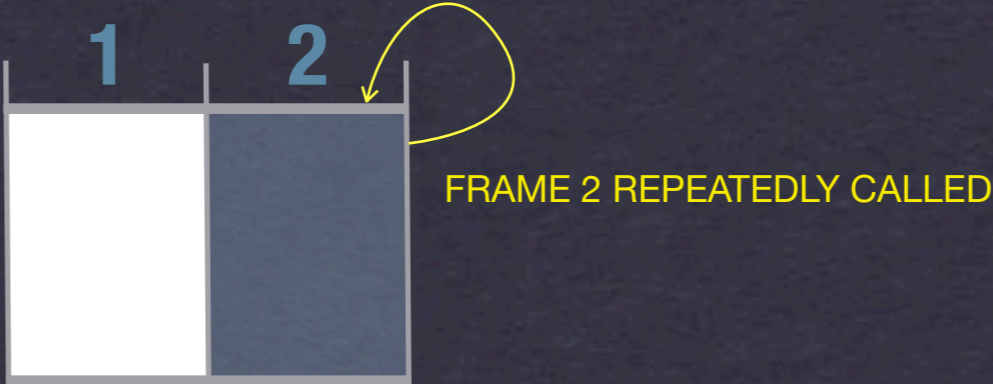
# FLASH SWF

TIMELINE



# FLEX SWF

TIMELINE



FRAME 1

FRAME 2

BOOTSTRAPPING/  
PRELOADER

APPLICATION CODE

# WHAT THE SWF KIND OF LOOKS LIKE

ARTIST (NOT-SO-MUCH) RENDERING

# The Problem

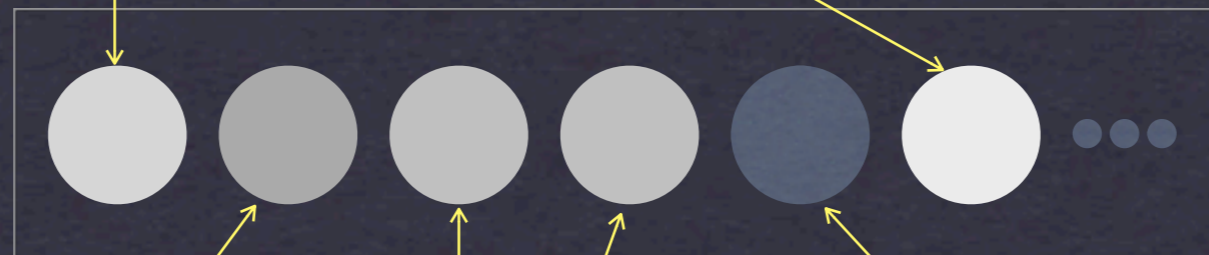
- ✱ Flash platform is event-driven: most everything is triggered by an event.
- ✱ Events collected into the Event Queue.
- ✱ FP handles one event at time.
- ✱ To get to the next frame? An event is triggered.

# TIMELINE



EVENTS DISPATCHED IN FRAME 1

# EVENT QUEUE



KEYBOARD EVENT

MOUSE EVENTS

**\*FRAME EVENT**

# THE EVENT QUEUE

ARTIST (NOT-SO-MUCH) RENDERING

# The Problem

- ✱ SWF plays at a certain number of frames per second (FPS). The time per frame is  $1/\text{FPS}$ .
- ✱ Default FPS is 24. Therefore, a frame needs to execute every ~42 milliseconds.
- ✱ Application FPS is not a minimum, it's a maximum.
- ✱ If code running in a frame takes longer than allotted time, FP cannot process next frame event "on time."
- ✱ Hello, non-responsive window/beach ball of death.

# The Problem

- ✱ Another way to look at it, using Ted Patrick's Elastic Mental Racetrack:

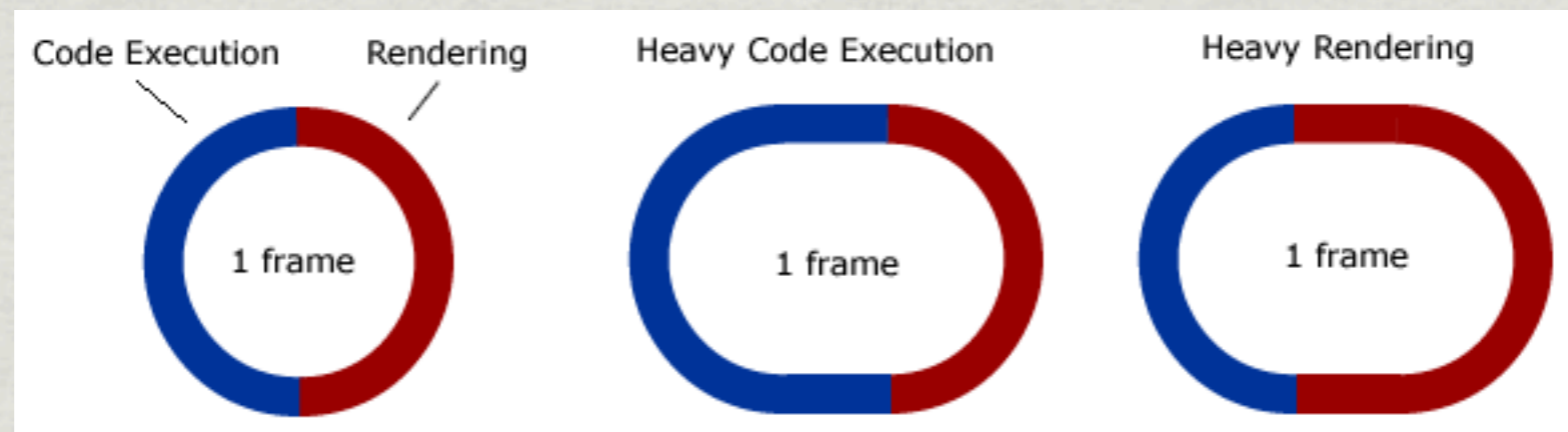


IMAGE FROM SEAN CHRISTMANN'S [UPDATED 'ELASTIC RACETRACK' FOR FLASH 9 AND AVM2](#)

- ✱ The racetrack extends to accommodate whatever code it is asked to execute.
- ✱ No built-in way to defer execution till later.
- ✱ If FP starts executing long-running code, code execution half will just keep elongating.
- ✱ FP gets stuck running on the first half of the track.

**GREENTHREADING IN FLEX**

# Some Solutions

- \* General idea: break up long-running code into smaller batches/jobs.
- \* Trigger batches at different times, different frames.
- \* Gives FP a chance to catch up.
- \* Allow other events to be processed (including the important screen-updating events).

# Some Solutions

- \* Several ways to break up the process:
  - \* `callLater()`: function called using this will be queued and run either at handler for RENDER event or at the next ENTER\_FRAME event
    - \* UIComponent method
    - \* *“Some features, like effects, can cause queued functions to be delayed until the feature completes...”*
  - \* ENTER\_FRAME: run a batch explicitly at each frame
  - \* Timers: run a batch every  $x$  milliseconds

# Some Solutions

## AND NOW FOR A LITTLE CODE...

```
// To break up an algorithm via callLater()  
// 1. Write a method that performs the main work of algorithm and takes as  
// parameters the current iteration and the total iterations to execute.  
// 2. In method heck for stopping condition and if it has not been reached,  
// use callLater to schedule a new call to method.  
  
...  
<!-- Some object that is being displayed -->  
<mx:UIComponent id="obj"/>  
  
...  
var numIterations : int = 0;  
var totalIterations : int = 100000;  
  
private function runLoop(... args) : void  
{  
    // Do work of iteration here.  
    numIterations++;  
    if(numIterations < totalIterations)  
    {  
        obj.callLater( runLoop, /* args to pass to next method call */ );  
    }  
}
```

GREENTHREADING IN FLEX

# Some Solutions

## AND NOW FOR A LITTLE CODE...

```
// To break up an algorithm via a ENTER_FRAME event.  
// 1. Initialize progress variables.  
// 2. Add a listener to the application for the ENTER_FRAME event the does  
//     main work of the algorithm.  
// 3. In handler check for stopping condition and remove the listener if it  
//     has been reached.  
  
var numIterations : int = 0;  
var totalIterations : int = 100000;  
Application.application.addEventListener(Event.ENTER_FRAME, runLoop);  
  
private function runLoop(event : Event) : void  
{  
    // Do work of iteration here.  
    numIterations++;  
    if(numIterations >= totalIterations)  
    {  
        Application.application.removeEventListener(Event.ENTER_FRAME, runLoop);  
    }  
}
```

GREENTHREADING IN FLEX

# Some Solutions

## AND NOW FOR A LITTLE CODE...

```
// To break up an algorithm via a Timer  
// 1. Initialize progress variables.  
// 2. Initialize a Timer with a delay. Here delay ≈ time/frame.  
// 3. Add listener for TimerEvent.TIMER that does main work of algorithm.  
// 4. In handler check for stopping condition and call stop() on Timer if it  
// has been reached.  
  
var numIterations : int = 0;  
var totalIterations : int = 100000;  
var timer : Timer = new Timer(1000 / Application.application.stage.frameRate);  
timer.addEventListener(TimerEvent.TIMER, runLoop);  
timer.start();  
  
private function runLoop(event : TimerEvent) : void  
{  
    // Do work of iteration here.  
    numIterations++;  
    if(numIterations >= totalIterations)  
    {  
        timer.stop();  
    }  
    // Else next iteration will execute next TimerEvent.TIMER event.  
}
```

GREENTHREADING IN FLEX

# Some Solutions

- \* The good:
  - \* Event Queue can process other events between batches.
  - \* Application remains responsive.

# Some Solutions

- \* The not-so-good:
  - \* Code becomes little more complex.
  - \* Can take much longer to finish the job:
    - \* Spread out over time.
    - \* More overhead.
  - \* Fine tune amount of work done and how often you do it.
    - \* Can be time-consuming for developer.
    - \* Might need to re-tune if code or data changes.

# Greenthreading

- \* Multi-threading on FP:
  - \* Developers can't access threading FP uses:
    - \* Networking
    - \* Watchers on your code for timeout
  - \* True threading available via PixelBender:
    - \* Designed for media processing: take advantage of ability of shaders, filters, and blenders to run multi-threaded.
    - \* Works great if you can find a way to convert your problem space to PixelBender's.

**GREENTHREADING IN FLEX**

# Greenthreading

- \* So what is a “green thread?”
  - \* Want to give developer more control over when code gets executed.
  - \* Can emulate threads in developer code, i.e., green threads.
  - \* Similar to how other platforms provide threading even for single processor systems.
- \* General idea:
  - \* Do as much work as possible in frame.
  - \* Leave enough time for other events to process, for screen to update.

GREENTHREADING IN FLEX

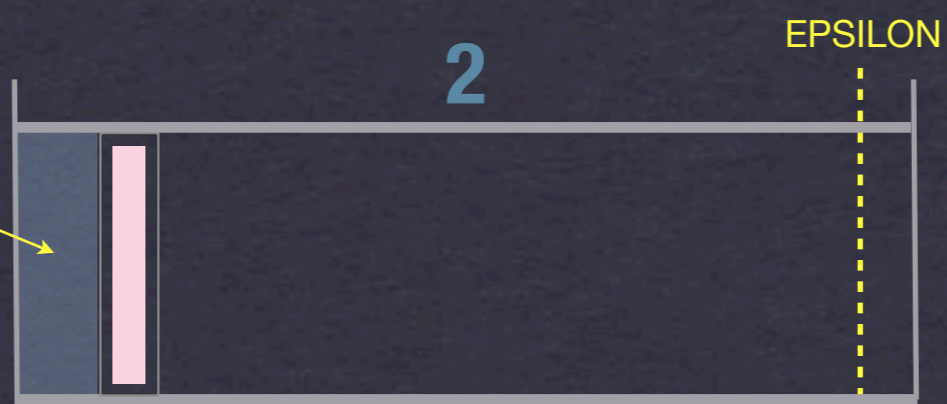
# Greenthreading

- \* How does it work?
  - \* Specify some amount of time to leave for other events: EPSILON
  - \* Break long-running code into basic blocks that will run repeatedly.
  - \* Each time greenthread finishes with a block, check how much total time used so far in frame.
  - \* If the total time used  $\geq$  `TIME_PER_FRAME - EPSILON`, stop running for now.

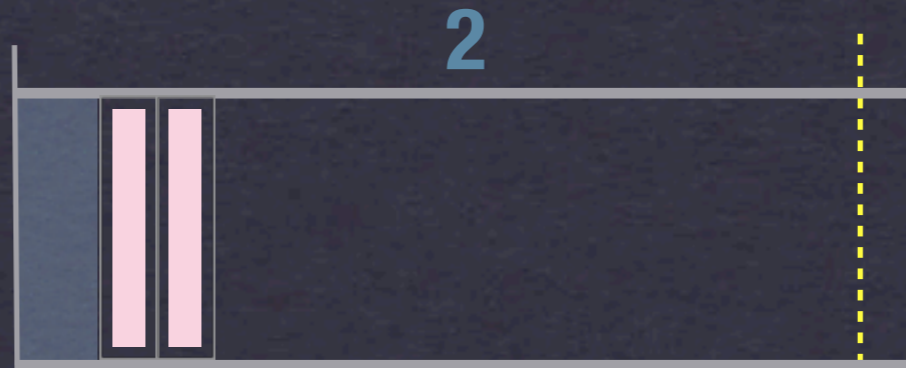
**GREENTHREADING IN FLEX**

EXECUTED BEFORE  
GREENTHREAD STARTS

AFTER 1 ITERATIONS



AFTER 2 ITERATIONS



AFTER X ITERATIONS



WAITING FOR NEXT FRAME



EXECUTED AFTER  
GREENTHREAD PAUSES

# GREENTHREADING

ARTIST (NOT-SO-MUCH) RENDERING

# Greenthreading

- \* Encapsulate all this as a class: GreenThread by Charlie Hubbard.
- \* <http://code.google.com/p/greenthreads/>
- \* Hubbard's class also provides statistics, progress monitoring via events.
- \* To use: convert your code block/algorithm/job etc. to a GreenThread.

# Greenthreading

AND NOW FOR A LITTLE CODE...

```
public class MyGreenThreadAlgorithm extends GreenThread
{
    // Called when start() called on a GreenThread
    override public function initialize() : void
    {
        // Initialize variables needed.
        progress = 0;
        maximum = 100;
    }

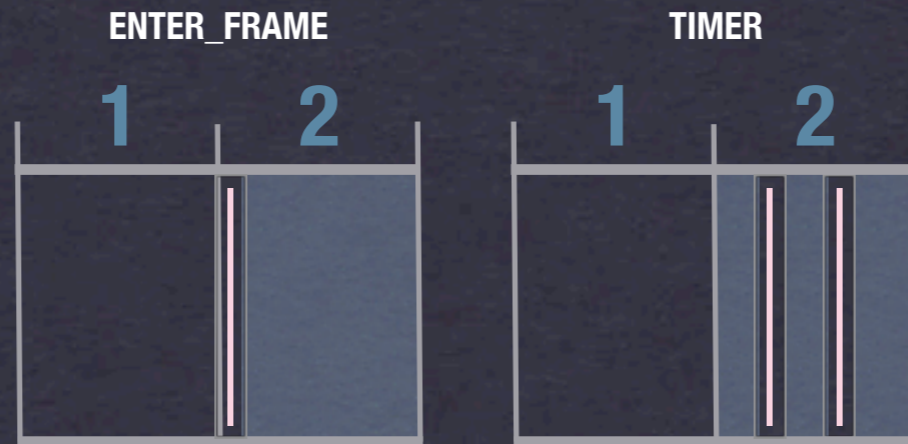
    override public function run() : Boolean
    {
        // Do work of one iteration here.
        progress++;
        // Return false if done, true otherwise.
    }
}
```

GREENTHREADING IN FLEX

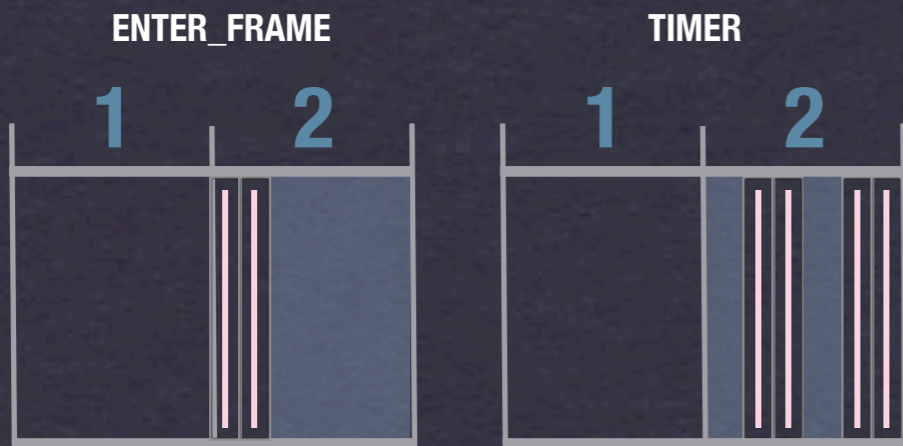
# ALGORITHM



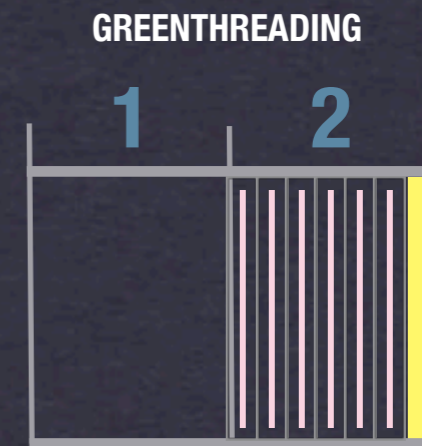
# DIVIDE + CONQUER



INCREASE ITERATIONS  
RUN PER FRAME



SPECIFY **EPSILON**: SET AMOUNT  
OF TIME LEFT IN FRAME FOR  
PROCESSING OTHER EVENTS.  
FILL UP REMAINING FRAME  
TIME WITH ITERATIONS



# GREENTHREADING VS OTHER SOLUTIONS

ARTIST (NOT-SO-MUCH) RENDERING, NOT DRAWN TO SCALE

	Original	Timer (1 loop/frame)	Timer (20k loop/frame)	GreenThread
<b>Runtime</b>	58 sec*	~11 hrs -	62 sec	85 sec
<b>Slow Down</b>	None*	V. Large	Small	Moderate
<b>Responsive ?</b>	NO	YES	NOT REALLY	YES

\*If the application doesn't crash that is.

## DEMO COMPARISON

### RUNTIME AND RESPONSIVENESS

# Greenthreading

- \* The not-so-good:
  - \* Still have overhead.
    - \* More code complexity
    - \* Runtime increase
  - \* Still may need to make adjustments to time left over to rest of events.

# Greenthreading

- \* The good:
  - \* Application responsiveness
  - \* Greenthread determines how much work to do itself given bounds.
  - \* Greenthread encapsulates management of time/work. Makes some things neater.

# Greenthreading

- \* But what about multi-threading?
  - \* Important to maintain EPSILON time to handle rest of the execution and rendering of the frame.
  - \* Queue up Greenthreads as they are instantiated.
  - \* Time allocated to each Greenthread:  $(\text{time for frame} - \text{EPSILON}) / \# \text{ active Greenthreads}$ .
  - \* When Greenthreads finish executing, they are removed from queue.
  - \* Implemented in Hubbard's library: just create as many GreenThreads as needed.

**GREENTHREADING IN FLEX**

# Credit Where Credit Is Due

- \* Doug Knudsen: “AIR, CSVs, and Mean Greenies oh my!” @ [cubicleman.com](http://cubicleman.com)
- \* Charlie Hubbard, “Actionscript and Concurrency” @ [wrongnotes.blogspot.com](http://wrongnotes.blogspot.com)
- \* Drew Cummins, “Green Threads” @ [blog.generalrelativity.org](http://blog.generalrelativity.org)
- \* Jesse Warden, “Parsing & Rendering Lots of Data in Flash Player” @ [jessewarden.com](http://jessewarden.com)

# Conclusion

- \* Applications that have long-running code can hang because Event Queue stalls.
- \* To keep application responsive:
  - \* Break work down into small jobs
  - \* Trigger jobs at different time, give Event Queue a chance to process other events.
- \* Greenthreading: fit as much work into frame as possible, leave time for other events.
- \* Costs of greenthreading: more complex code, little longer runtime.

# Questions?

**THANKS FOR COMING!**

**HUYEN TUE DAO**

**DAOTUEH@GMAIL.COM**

**QUEENCODEMONKEY @ TWITTER**

**WWW.QUEENCODEMONKEY.COM**

**GREENTHREADING IN FLEX**

# Links

- \* Greenthreading

- \* <http://www.cubicleman.com/2009/03/08/air-csvs-and-mean-greenies-oh-my/>

- \* <http://wrongnotes.blogspot.com/2009/02/concurrency-and-actionscript-part-i-of.html>

- \* <http://blog.generalrelativity.org/actionscript-30/green-threads/>

- \* <http://jessewarden.com/2009/02/parsing-rendering-lots-of-data-in-flash-player.html>

- \* Frame execution model

- \* <http://www.craftymind.com/2008/04/18/updated-elastic-racetrack-for-flash-9-and-avm2/>

- \* <http://www.onflex.org/ted/2005/07/flash-player-mental-model-elastic.php>