INSTITUTO NOKIA DE TECNOLOGIA

Considering Tiling for a better User Experience

Kenneth Rohde Christiansen
WebKit Code Camp, Wiesbaden, December 2009

openBossa
Software in a creative way

SOMETHING THAT I HAVE BEEN WORKING ON

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I?

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I? KENNETH R. CHRISTIANSEN

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I? KENNETH R. CHRISTIANSEN WEBKIT REVIEWER

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I? KENNETH R. CHRISTIANSEN WEBKIT REVIEWER
1+ YEAR WORKING WITH WEBKIT

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I? KENNETH R. CHRISTIANSEN WEBKIT REVIEWER
1+ YEAR WORKING WITH WEBKIT 3+ YEARS WORKING WITH
APP & FRAMEWORK DEVELOPMENT AT

SOMETHING THAT I HAVE BEEN WORKING ON

WHO AM I? KENNETH R. CHRISTIANSEN WEBKIT REVIEWER 1+ YEAR WORKING WITH WEBKIT 3+ YEARS WORKING WITH APP & FRAMEWORK DEVELOPMENT AT NOKIA TECHNOLOGY INSTITUTE, INdT, RECIFE, BRAZIL

A TILED

BACKING STORE

What are the problems we are facing today?

The background for considering tiling

What are the problems we are facing today?

The background for considering tiling

WE PAINT THINGS WE'VE ALREADY PAINTED WHEN SCROLLING BACK

What are the problems we are facing today?

The background for considering tiling

WE PAINT THINGS WE'VE ALREADY PAINTED WHEN SCROLLING BACK

WE CALL INTO WEBCORE FOR EACH PAINT...

What are the problems we are facing today?

The background for considering tiling

WE PAINT THINGS WE'VE ALREADY PAINTED WHEN SCROLLING BACK

WE CALL INTO WEBCORE FOR EACH PAINT...

...CALLING INTO WEBCORE HAS SOME OVERHEAD

What are the problems we are facing today?

The background for considering tiling

WE PAINT THINGS WE'VE ALREADY PAINTED WHEN SCROLLING BACK

WE CALL INTO WEBCORE FOR EACH PAINT...

FOR INSTANCE, CONSTRUCTING GRAPHICS CONTEXT IS QUITE EXPENSIVE

...CALLING INTO WEBCORE HAS SOME OVERHEAD

What are the problems we are facing today?

The background for considering tiling

WE PAINT THINGS WE'VE ALREADY PAINTED WHEN SCROLLING BACK

WE CALL INTO WEBCORE FOR EACH PAINT...

FOR INSTANCE, CONSTRUCTING GRAPHICS CONTEXT IS QUITE EXPENSIVE

...CALLING INTO WEBCORE HAS SOME OVERHEAD

CLEVER TILING CAN SOLVE THESE ISSUES

How to accomplish this

What is tiling anyway?

Cache and join paint events

How to accomplish this

What is tiling anyway?

Cache and join paint events

Cache what you paint in image tiles

How to accomplish this

What is tiling anyway?

Cache and join paint events

Cache what you paint in image tiles

Blit the existing tiles on scroll

How to accomplish this

What is tiling anyway?

Cache and join paint events

Cache what you paint in image tiles

Blit the existing tiles on scroll

Don’t paint non-visible dirty areas immediately

How to accomplish this

What is tiling anyway?

Cache and join paint events

Cache what you paint in image tiles

Blit the existing tiles on scroll

Don’t paint non-visible dirty areas immediately

Avoid too many small tiles, due to the cost of constructing GraphicsContexts

How to accomplish this

What is tiling anyway?

Cache and join paint events

Cache what you paint in image tiles

Blit the existing tiles on scroll

Don’t paint non-visible dirty areas immediately

Avoid too many small tiles, due to the cost of constructing GraphicsContexts

This can be hardware accelerated!

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

- Implemented only for QGraphicsWebView

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

- Implemented only for QGraphicsWebView
- Some changes needed elsewhere:
  1) Render methods in abs. coordinates, without clipping

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

- Implemented only for QGraphicsWebView
- Some changes needed elsewhere:
  1) Render methods in abs. coordinates, without clipping
  2) Make ScrollView / FrameView send update events outside of the viewport

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

- Implemented only for QGraphicsWebView
- Some changes needed elsewhere:
    1) Render methods in abs. coordinates, without clipping
    2) Make ScrollView / FrameView send update events outside of the viewport

Why not just make viewport == contents size?

My experiment

My basic algorithm

As an experiment, implement it on the WebKit side

- Implemented only for QGraphicsWebView
- Some changes needed elsewhere:
  1) Render methods in abs. coordinates, without clipping
  2) Make ScrollView / FrameView send update events outside of the viewport

Why not just make viewport == contents size?

Because we use WebCore for drawing our scrollbars and that makes a whole lot of sense ... theming ... etc

My experiment

My basic algorithm

Basic algorithm

My experiment

My basic algorithm

Basic algorithm

If not in cache, paint the dirty area as a tile, enlarge it slightly (64 pixels in each direction)

My experiment

My basic algorithm

Basic algorithm

- If not in cache, paint the dirty area as a tile, enlarge it slightly (64 pixels in each direction)
- Put in cache, blit to screen

My experiment

My basic algorithm

Basic algorithm

If not in cache, paint the dirty area as a tile, enlarge it slightly (64 pixels in each direction)

Put in cache, blit to screen

Each tile stores it's covered area as well as a dirty area.

My experiment

My basic algorithm

Basic algorithm

If not in cache, paint the dirty area as a tile, enlarge it slightly (64 pixels in each direction)

Put in cache, blit to screen

Each tile stores it’s covered area as well as a dirty area.

On update, we update the dirty area of the intersected tiles. If it has such an area already, the bounding rect is uses as the new area. Remember, we try to avoid calling into WebCore unnecessarily

My experiment

My basic algorithm

Basic algorithm

If not in cache, paint the dirty area as a tile, enlarge it slightly (64 pixels in each direction)

Put in cache, blit to screen

Each tile stores it's covered area as well as a dirty area.

On update, we update the dirty area of the intersected tiles. If it has such an area already, the bounding rect is uses as the new area. Remember, we try to avoid calling into WebCore unnecessarily

Furthermore, if the dirty area == covered area, remove it from cache

My experiment

My basic algorithm

Basic algorithm

My experiment

My basic algorithm

Basic algorithm

On update and scroll:

Update all tiles, blit what is in the cache, create tiles for what is not

My experiment

My basic algorithm

Basic algorithm

On update and scroll:

Update all tiles, blit what is in the cache, create tiles for what is not

That is more of less the basic algorithm, but there are some problems.

My experiment

My basic algorithm

Problems

My experiment

My basic algorithm

Problems

Scrollbars

Paint the scrollbars separately, make sure updates to them do not invalidate any tiles. These are not tiled in my implementation!

My experiment

My basic algorithm

Problems

Scrollbars

Paint the scrollbars separately, make sure updates to them do not invalidate any tiles. These are not tiled in my implementation!

Enlarging tiles

When enlarging we much make sure that we don't cover an area already in the cache, so I need to know what area is cached.

My experiment

My basic algorithm

Problems

My experiment

My basic algorithm

Problems

Cache growth

Storing the whole page in the cache would be expensive, memory-wise, so the cache has a max size.

My experiment

My basic algorithm

Problems

Cache growth

Storing the whole page in the cache would be expensive, memory-wise, so the cache has a max size.

The solution is to give each tile an age and increase the age when not used, and reset it when being blit.

My experiment

My basic algorithm

Problems

Cache growth

Storing the whole page in the cache would be expensive, memory-wise, so the cache has a max size.

The solution is to give each tile an age and increase the age when not used, and reset it when being blit.

Before adding a new tile, we reserve the needed space for it, removing the oldest tiles.

So what is the preliminary results?

Did it really pay off?

Benjamin wrote a simple scrolling test app, and the results are quite promising.

So what is the preliminary results?

Did it really pay off?

Benjamin wrote a simple scrolling test app, and the results are quite promising.

QWEBVIEW PERFORMS RESONABLE, DUE TO THE XCOPYREGION

So what is the preliminary results?

Did it really pay off?

Benjamin wrote a simple scrolling test app, and the results are quite promising.

QWEBVIEW PERFORMS RESONABLE, DUE TO THE XCOPYREGION

THE TILING IS FASTER!

So what is the preliminary results?

Did it really pay off?

Benjamin wrote a simple scrolling test app, and the results are quite promising.

QWEBVIEW PERFORMS RESONABLE, DUE TO THE XCOPYREGION

THE TILING IS FASTER!

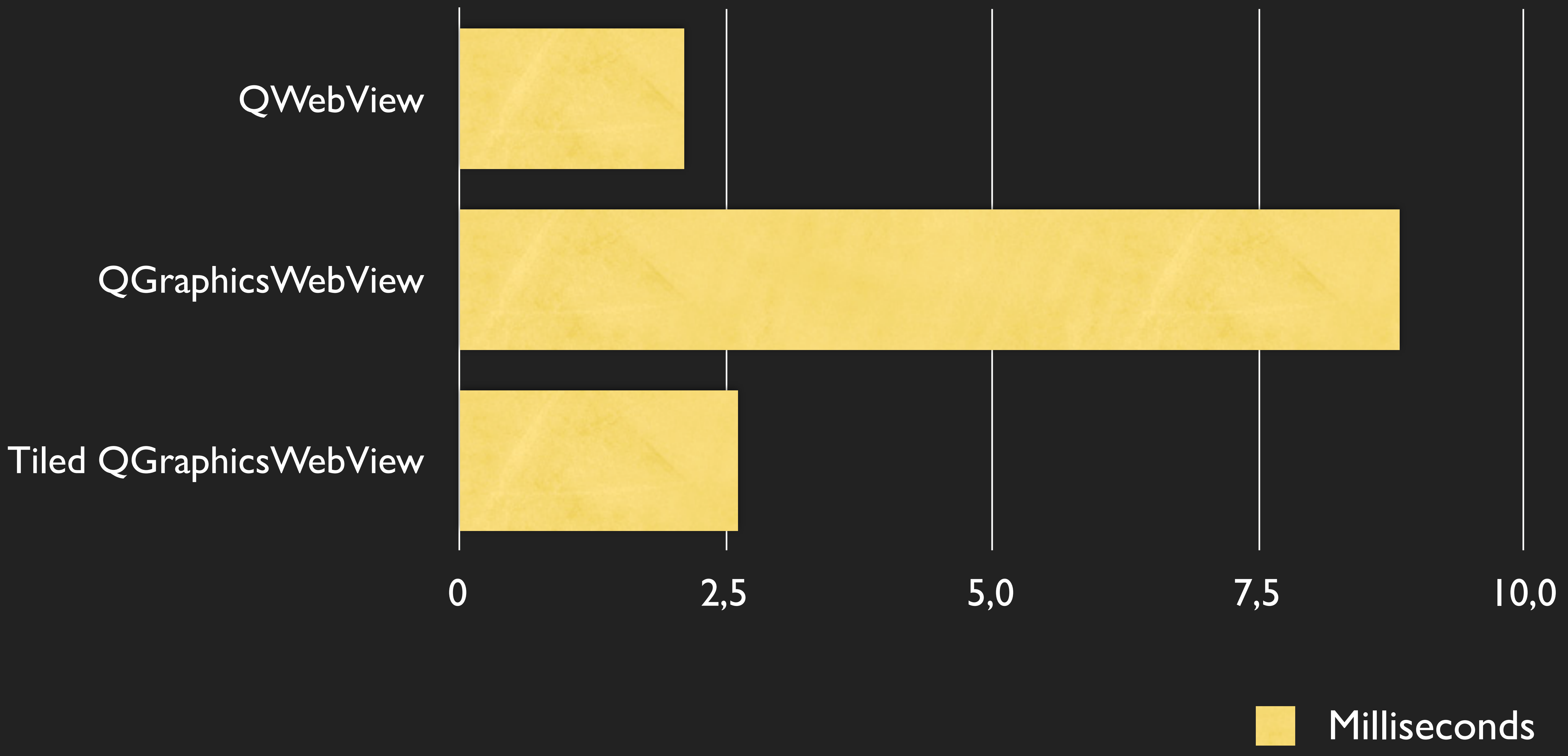
But we had some surprises as well.

NON-TILED QGRAPHICSWEBVIEW IS EXTREMELY SLOW

Where are se' numbers?

Back up your claims dude!

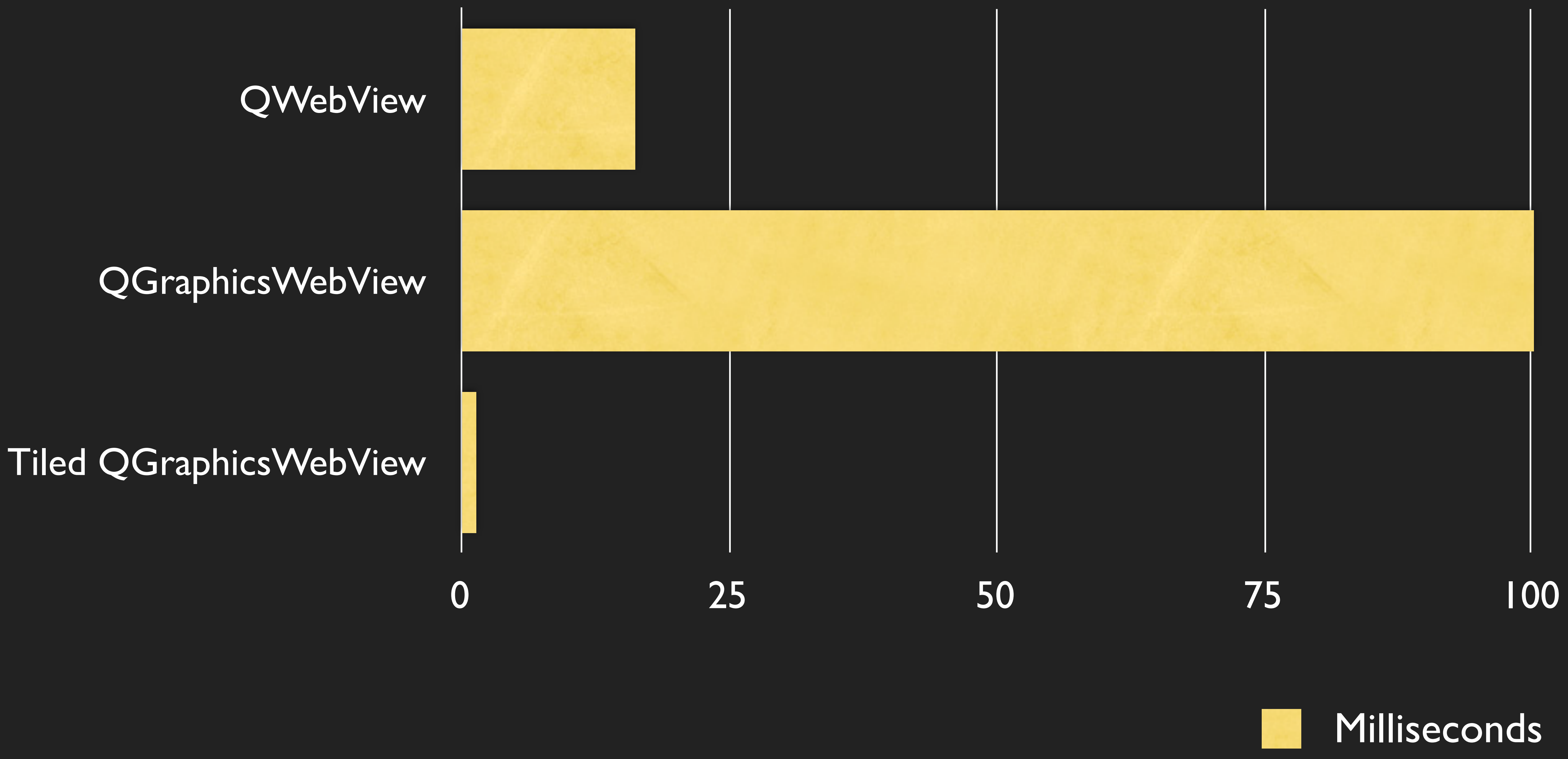
Simple painting



Where are se' numbers?

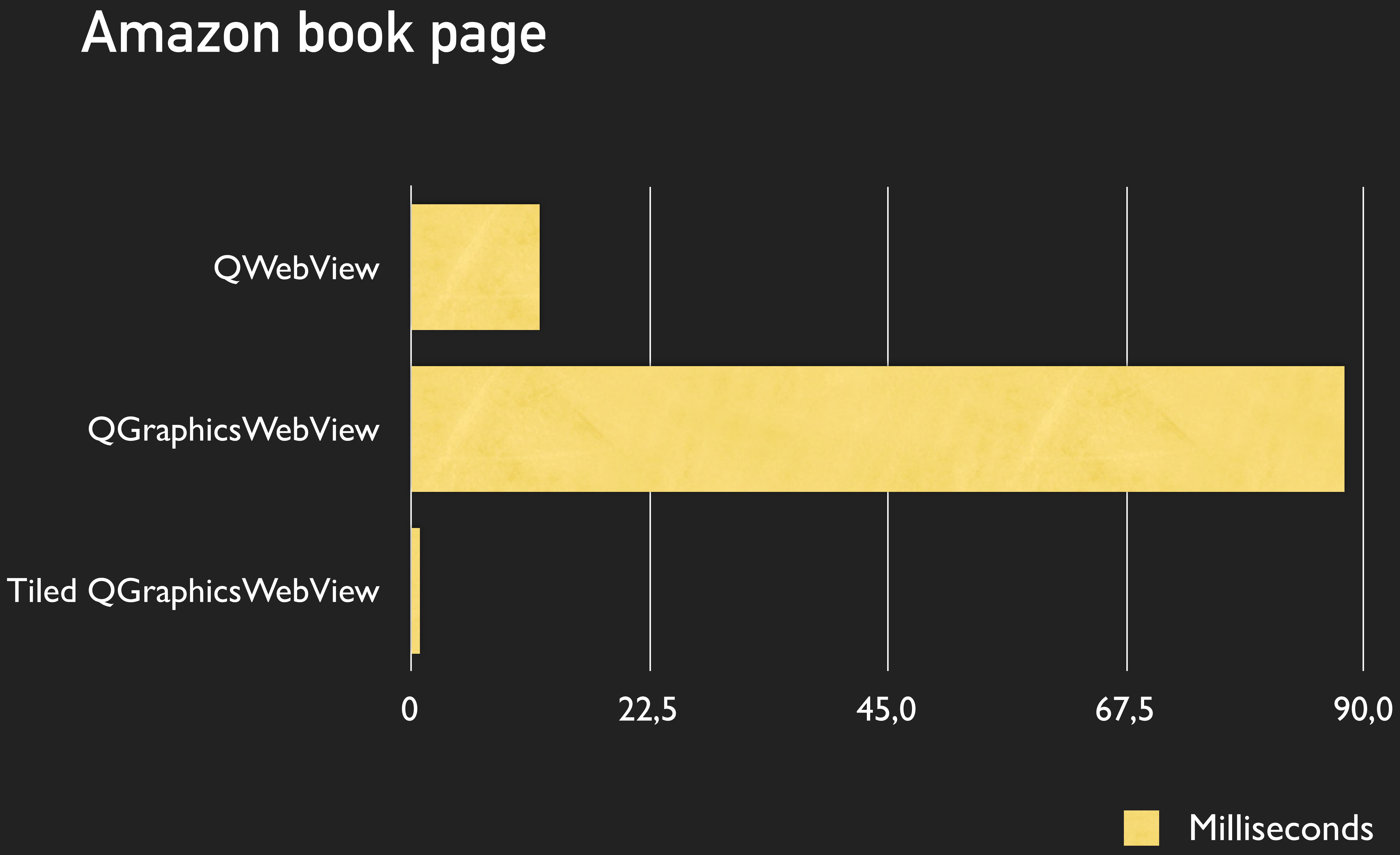
Back up your claims dude!

Slashdot page



Where are se' numbers?

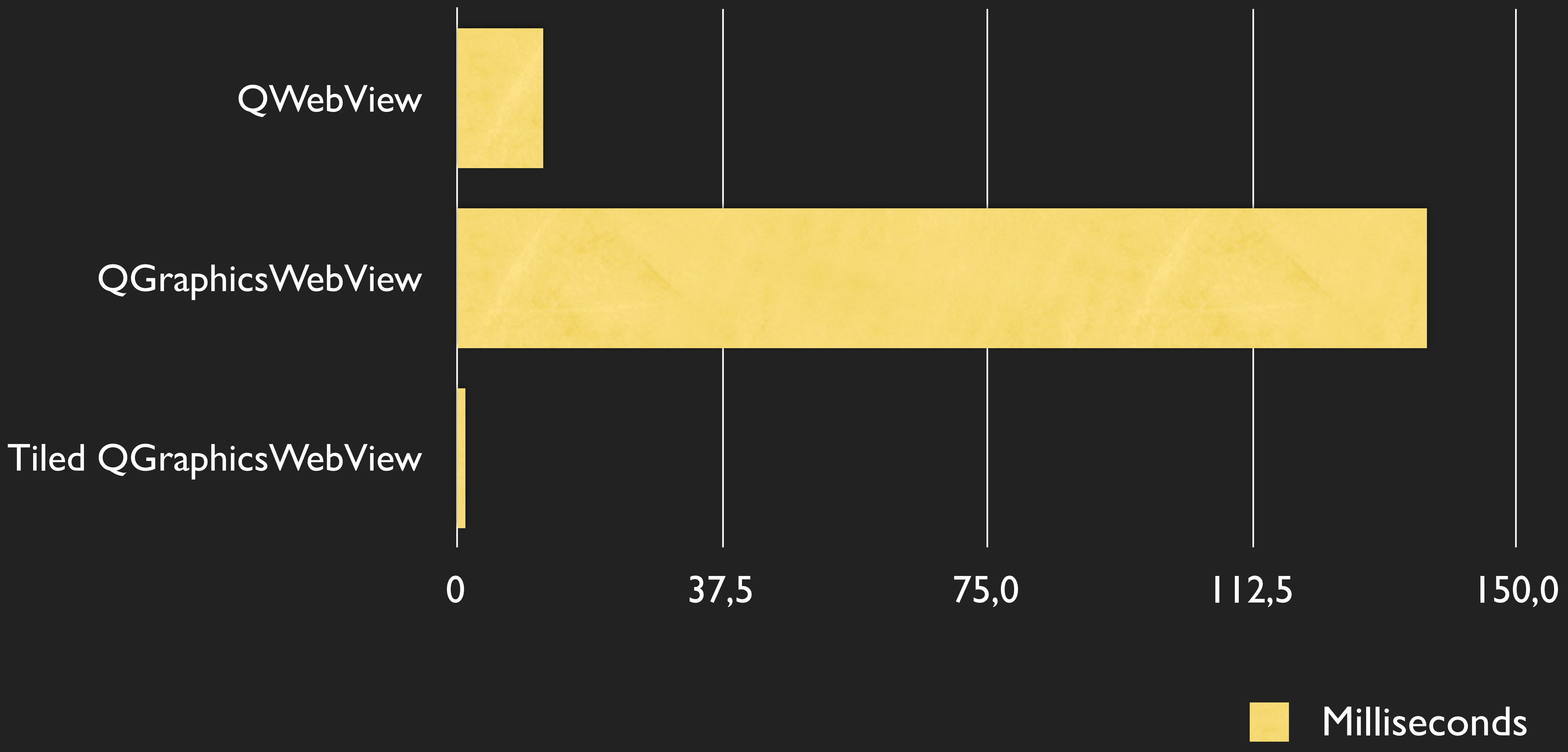
Back up your claims dude!



Where are se' numbers?

Back up your claims dude!

Wikipedia Qt page



Can’t you do better than that?

Ideas for improvement

WE CAN!

Can’t you do better than that?

Ideas for improvement

WE CAN!

More realistic test suite!

Can’t you do better than that?

Ideas for improvement

WE CAN!

- More realistic test suite!
- My data structures are not that good, nor profiled

Can't you do better than that?

Ideas for improvement

WE CAN!

More realistic test suite!

My data structures are not that good, nor profiled

Should be lower in the stack using WebCore constructs

Can’t you do better than that?

Ideas for improvement

WE CAN!

More realistic test suite!

My data structures are not that good, nor profiled

Should be lower in the stack using WebCore constructs

Paint in another thread, not block WebCore

Can’t you do better than that?

Ideas for improvement

WE CAN!

- More realistic test suite!
- My data structures are not that good, nor profiled
- Should be lower in the stack using WebCore constructs
- Paint in another thread, not block WebCore

That is why we are here ;-) Now let’s get on to the work!

Thanks for listening

KENNETH ROHDE CHRISTIANSEN

ext-kenneth.christiansen@nokia.com
kenneth.christiansen@openbossa.org