Rendering for "Glass Time"

Time usage in "rendering" and its affect on animation, jank and (if time) latency.

http://goo.gl/MHGWTc

Outcome

By the end of this talk you should understand;

- How the choice of time used in "rendering[1]" affects both **animation** and (if time) **input latency**.

- The concepts of **lag** and **glass time** and how it is used to reach a zero latency, jank free, silky smooth system.

And finally, as this is BlinkOn,

- How Chrome, Blink and the web currently work and need to adapt.

Experiments

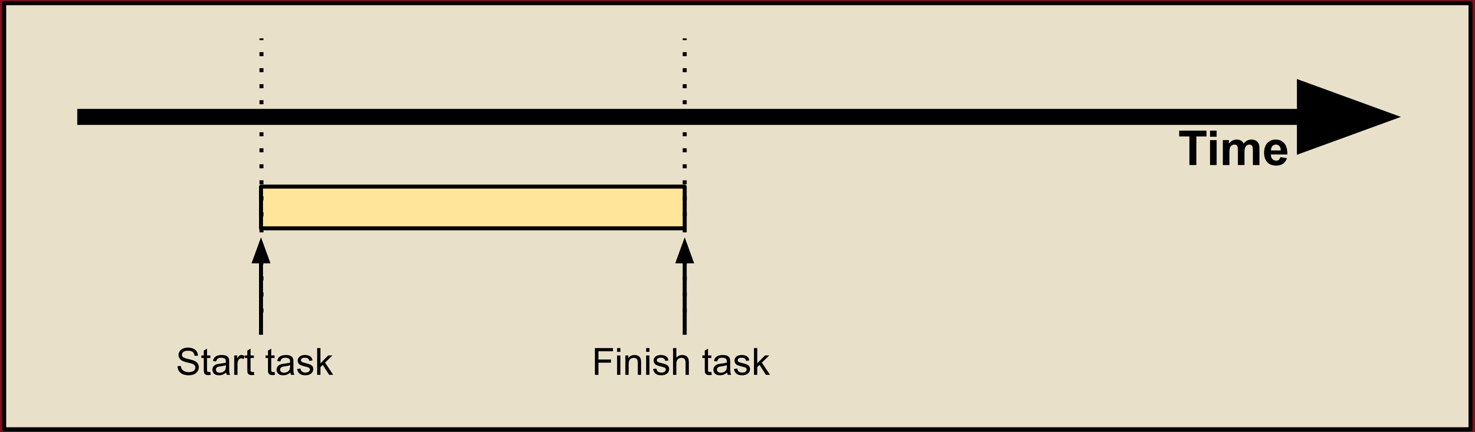
Two useful experiments we will use;

- Time Square Ball “NYE Drop”
- Under Finger Tracking Ball

Will be explained further later in the presentation.

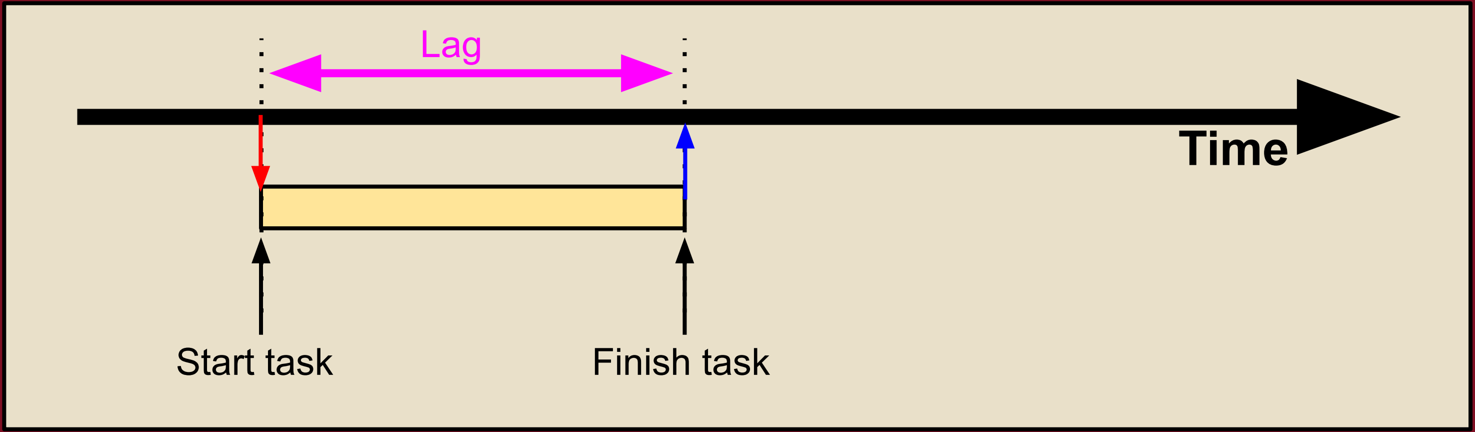
Things take time

Even in the world of super powerful computers, things are not instant.



This creates lag

When you have finished the task,
the world has already moved on!



http://goo.gl/MHGWTc

“Rendering”

Time taken to create pixels in a state suitable for output to the display.

For simplicity will think of “rendering” as a single task which takes time.

“Rendering”

In **Chrome** rendering is a complicated
and multi-stage process.

We can think of “rendering” as being roughly all of,

- Time in main thread - “Blink rendering”
- Time in impl thread - “Compositing”
- Time in getting to GPU - “Activation of trees”

Experiment 1

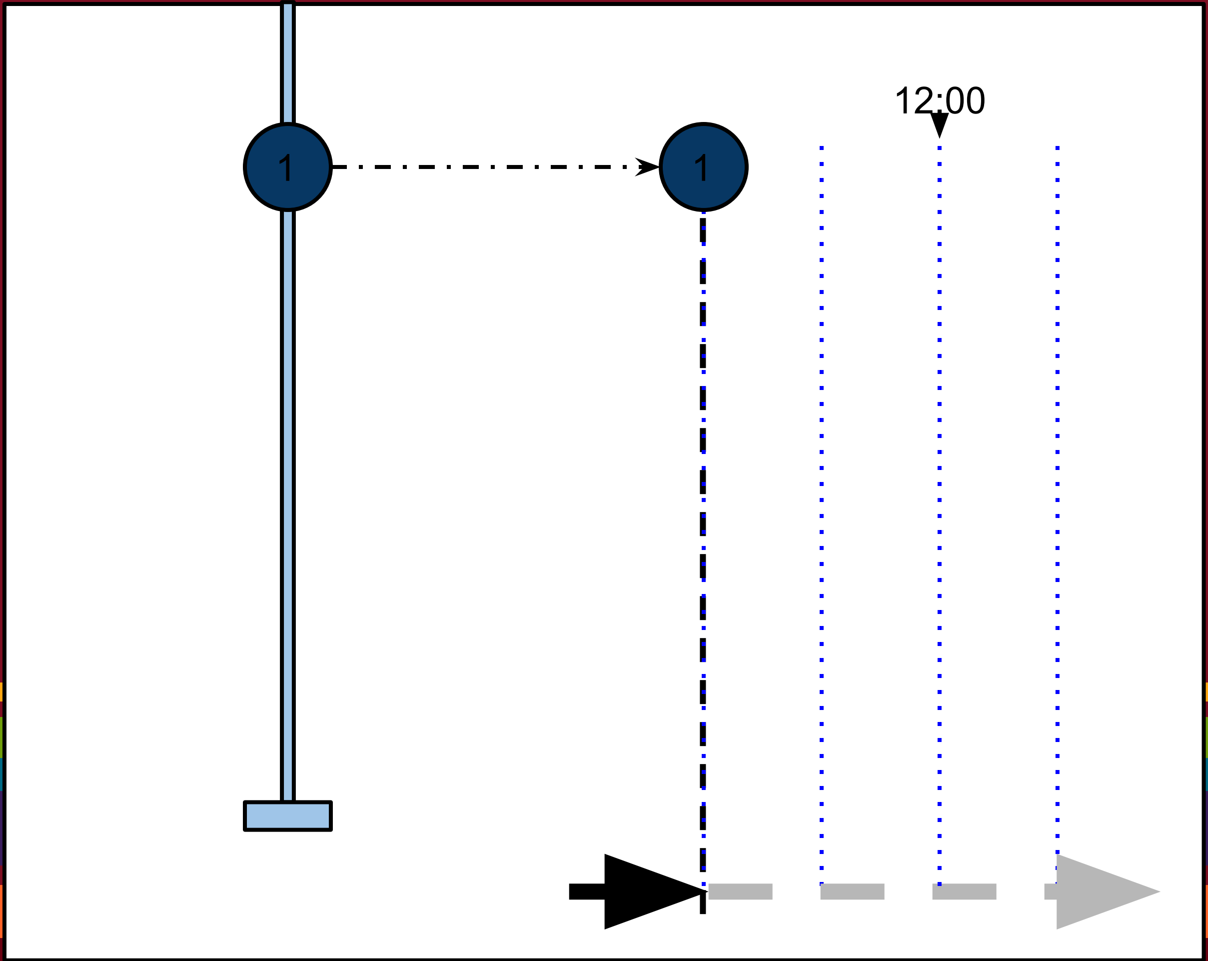
An example of how rendering time effects output.

Experiment 1 - Times Square Ball

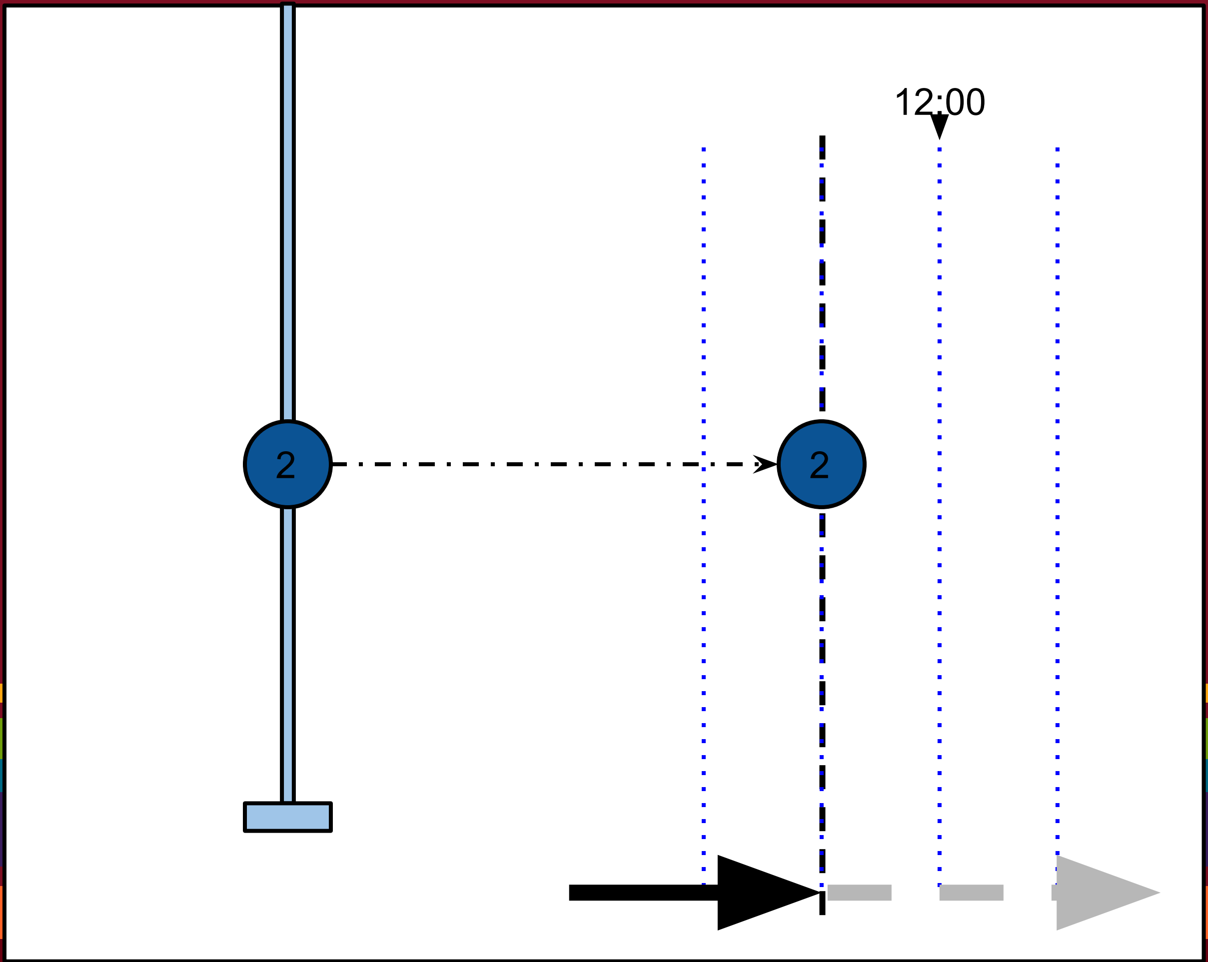
- Starts dropping **11:59**pm on NYE

- Finish at the bottom at **12:00** “starting” the NYE fireworks

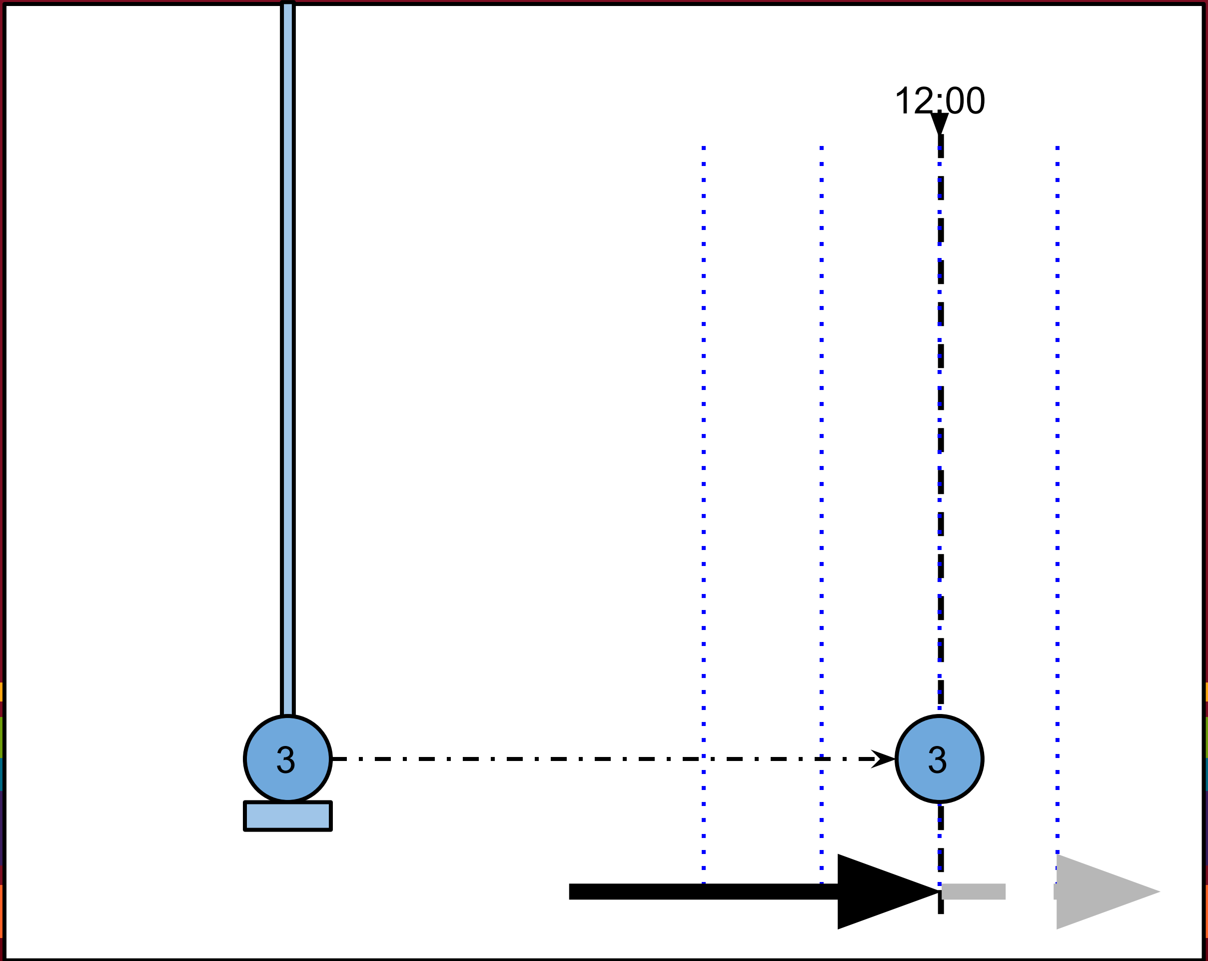
http://goo.gl/MHGWTc



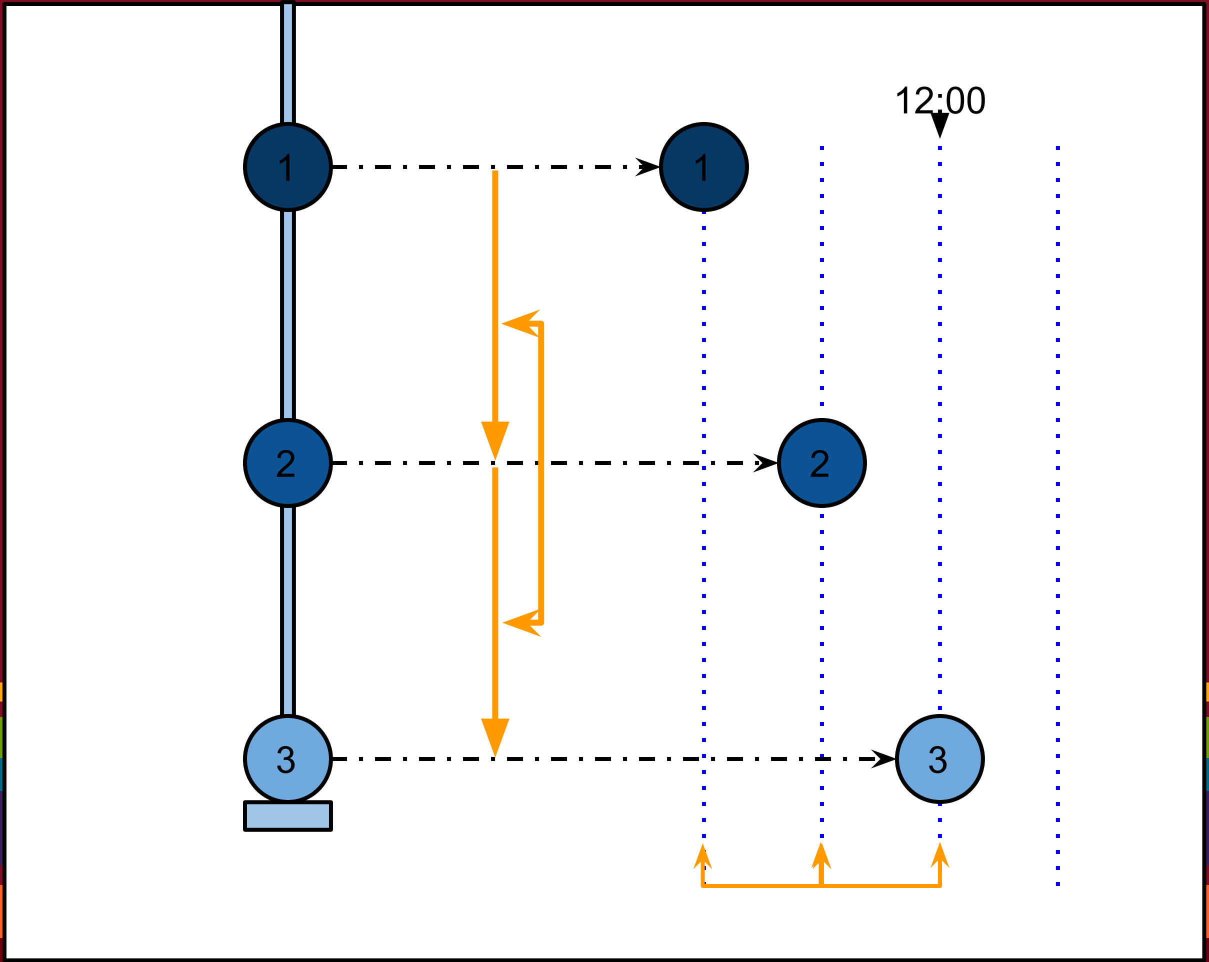
http://goo.gl/MHGWTc



http://goo.gl/MHGWTc



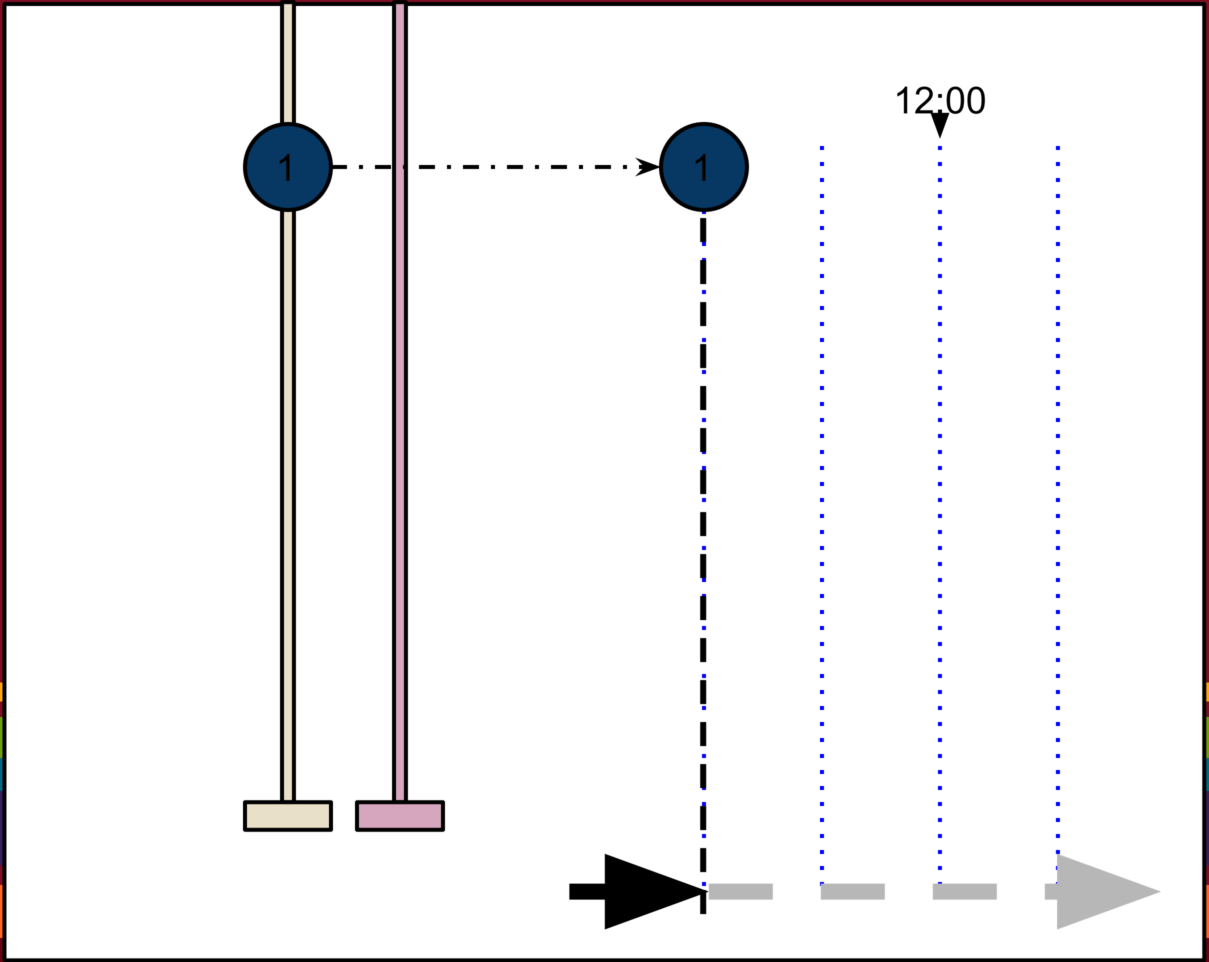
http://goo.gl/MHGWTc



http://goo.gl/MHGWTc

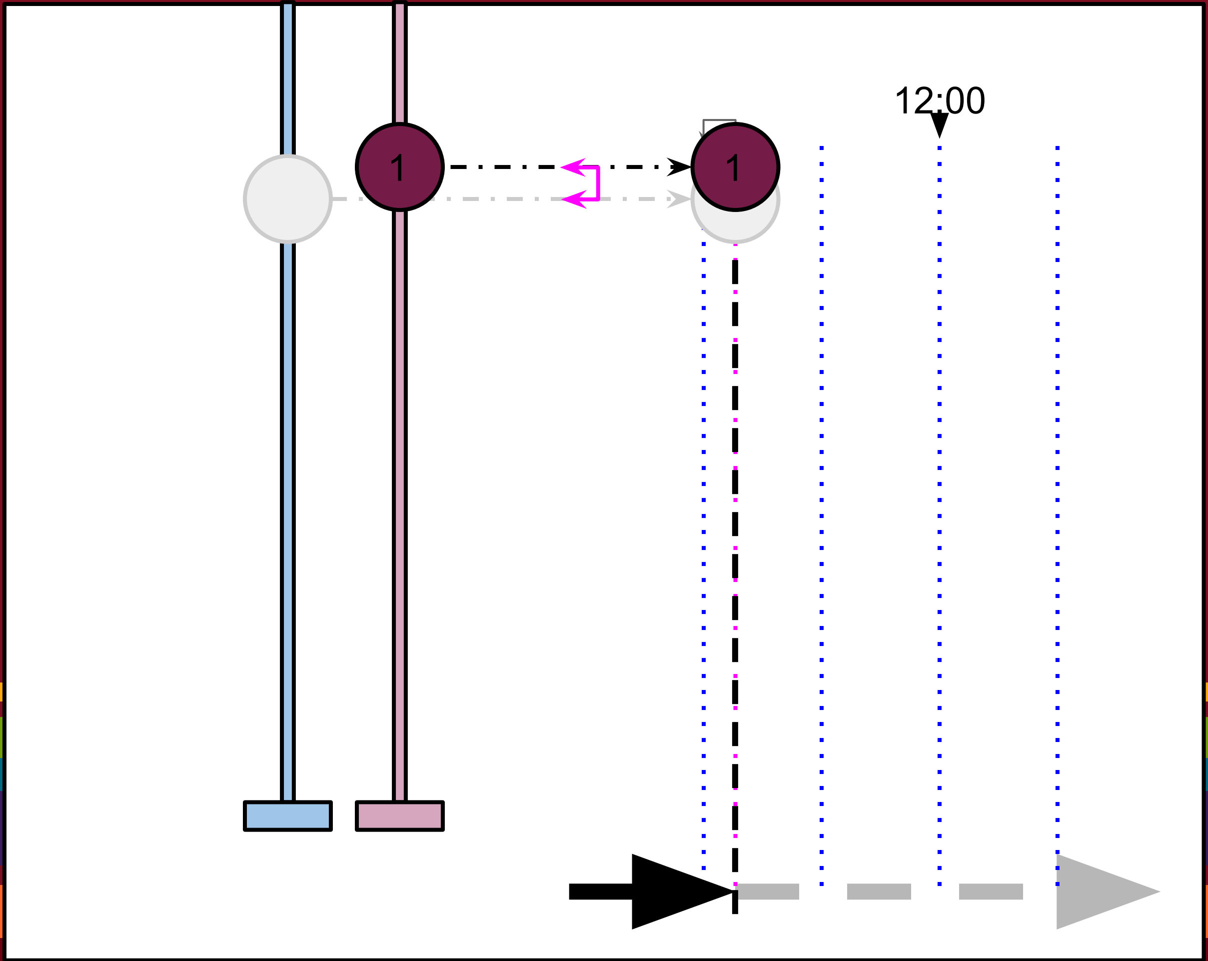
How does lag affect the system?

http://goo.gl/MHGWTc



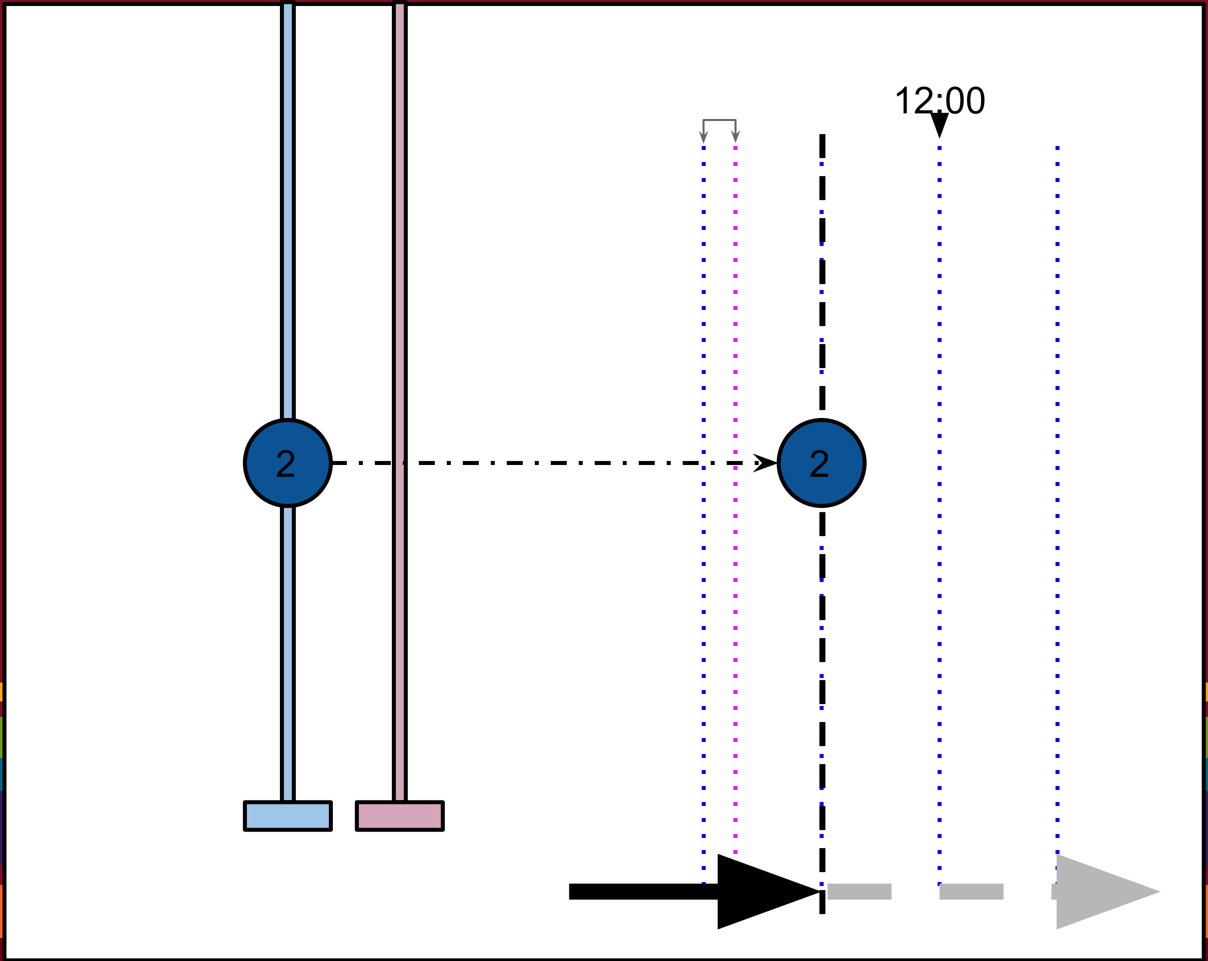
- Start “render”.

http://goo.gl/MHGWTc



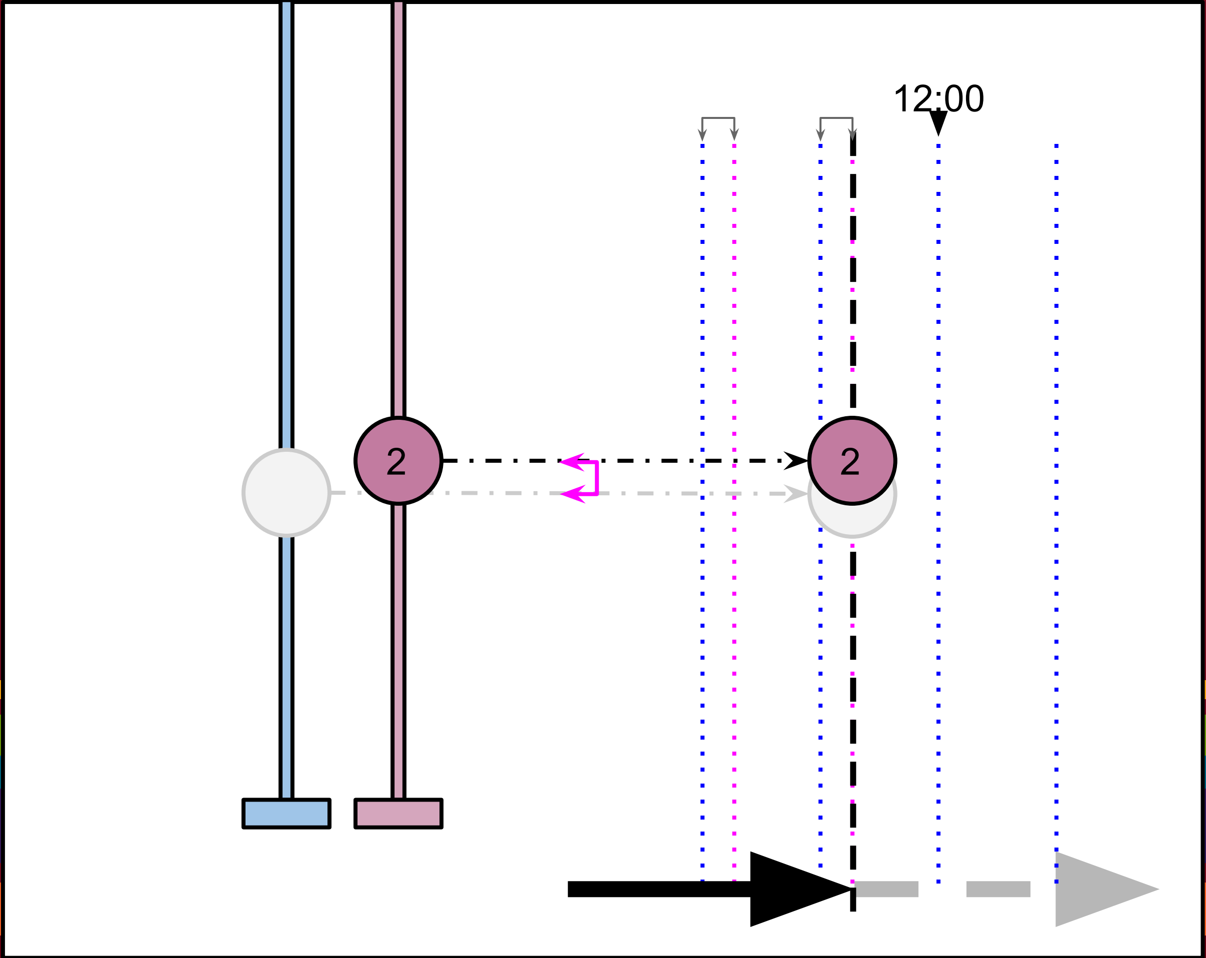
- End “render”.
- Real world has moved on.

http://goo.gl/MHGWTc



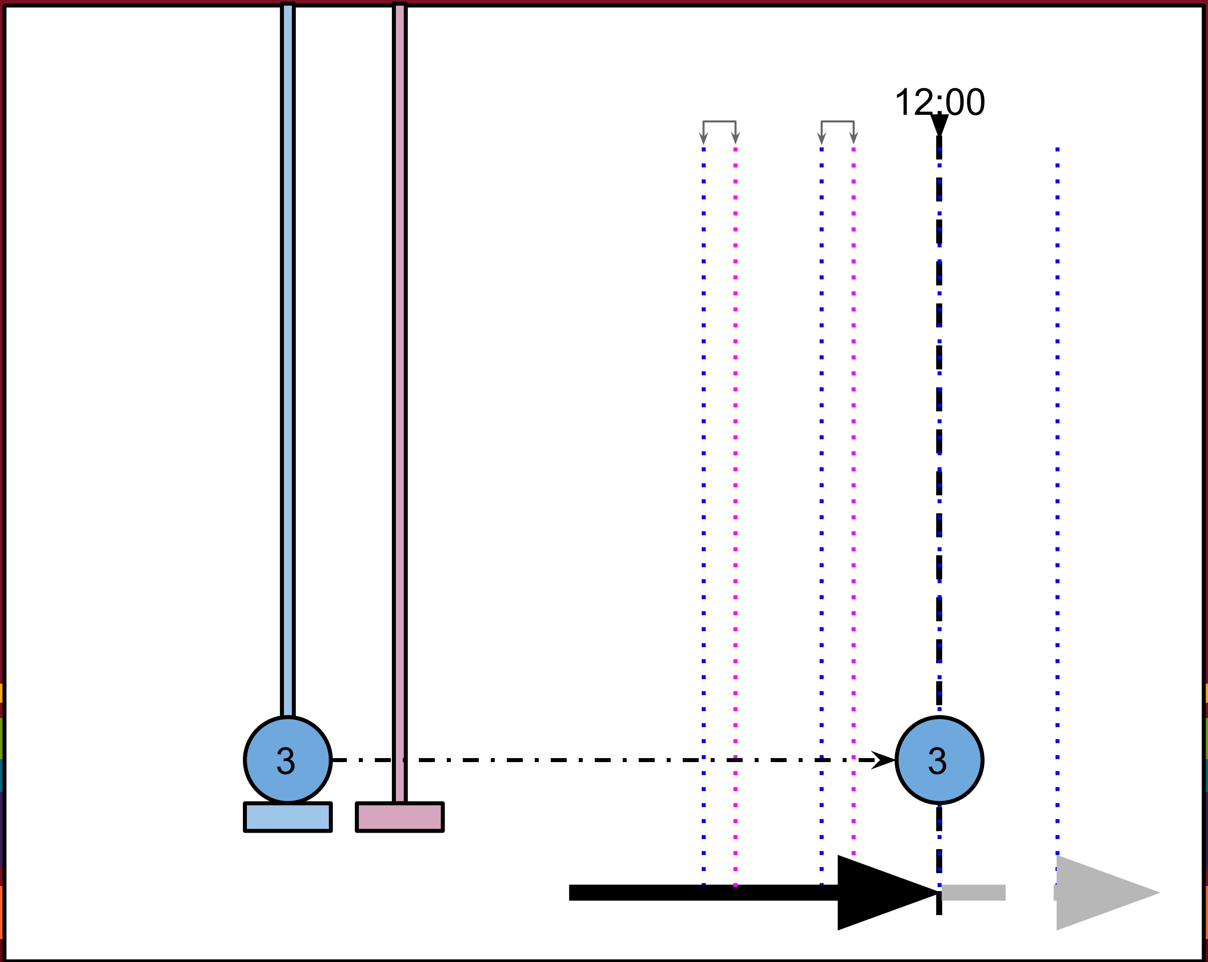
- Start "render".

http://goo.gl/MHGWTc



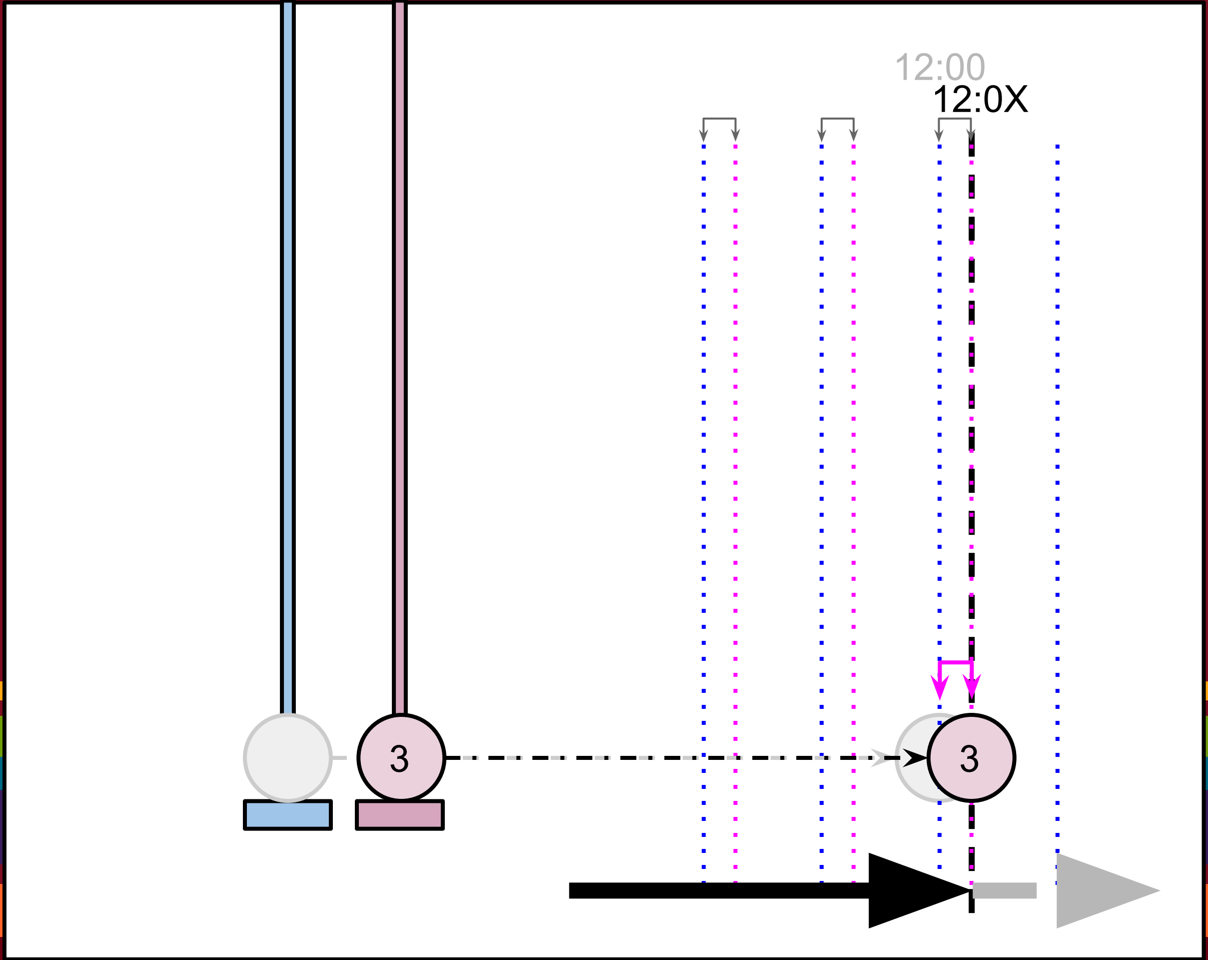
- End “render”.
- Real world has moved on.

http://goo.gl/MHGWTc



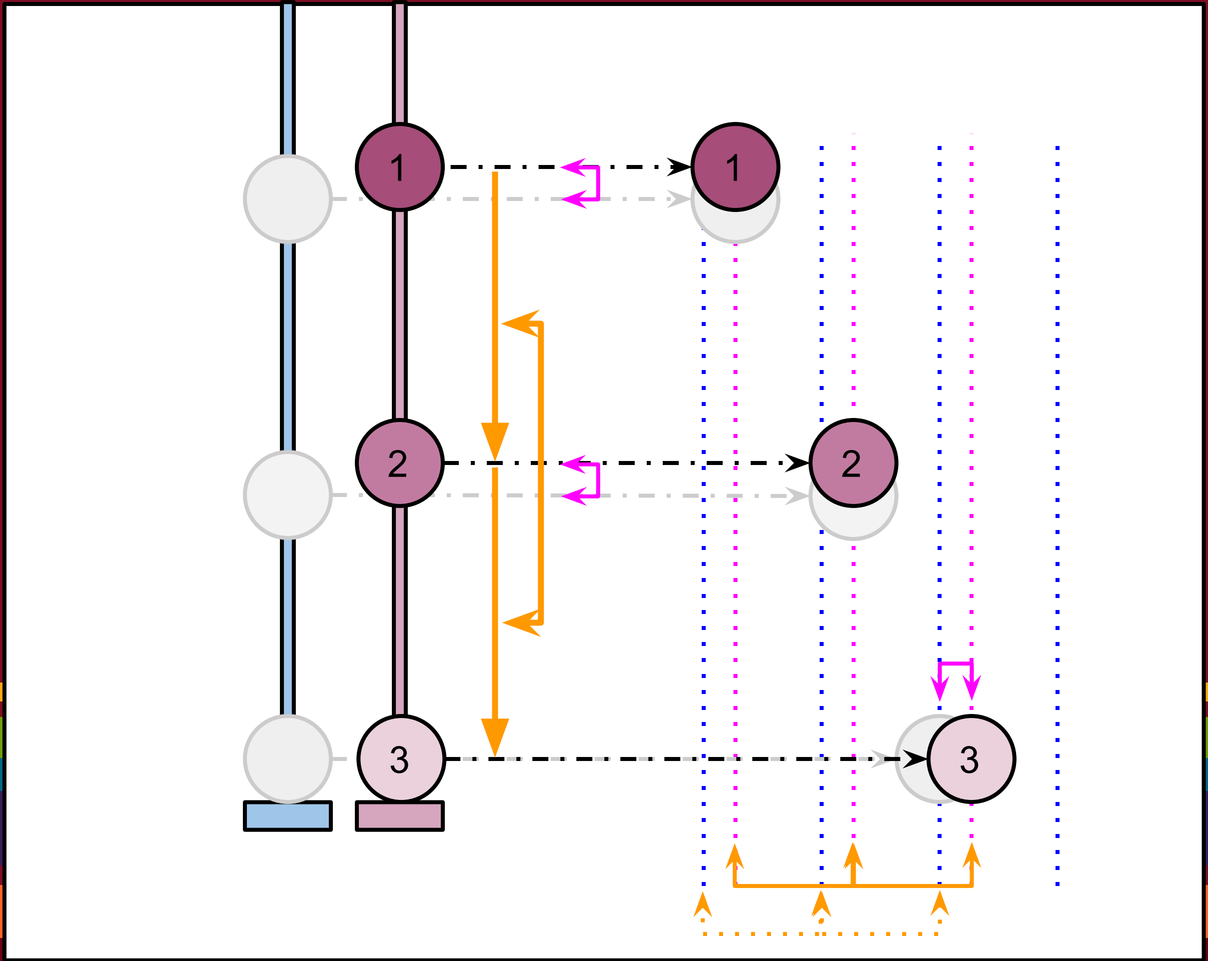
- Start “render”.
- Ball reaches **`y=0`**.
- Fireworks goes off.

http://goo.gl/MHGWTc



- End “render”.
- Simulation ball reaches output.
- Simulation was late :(

http://goo.gl/MHGWTc



- Constant "render time", gives constant lag error.
- (If "render time" wasn't constant, lag error wouldn't be constant.)

http://goo.gl/MHGWTc

http://goo.gl/MHGWTc

Are there other times to render for?

http://goo.gl/MHGWTc

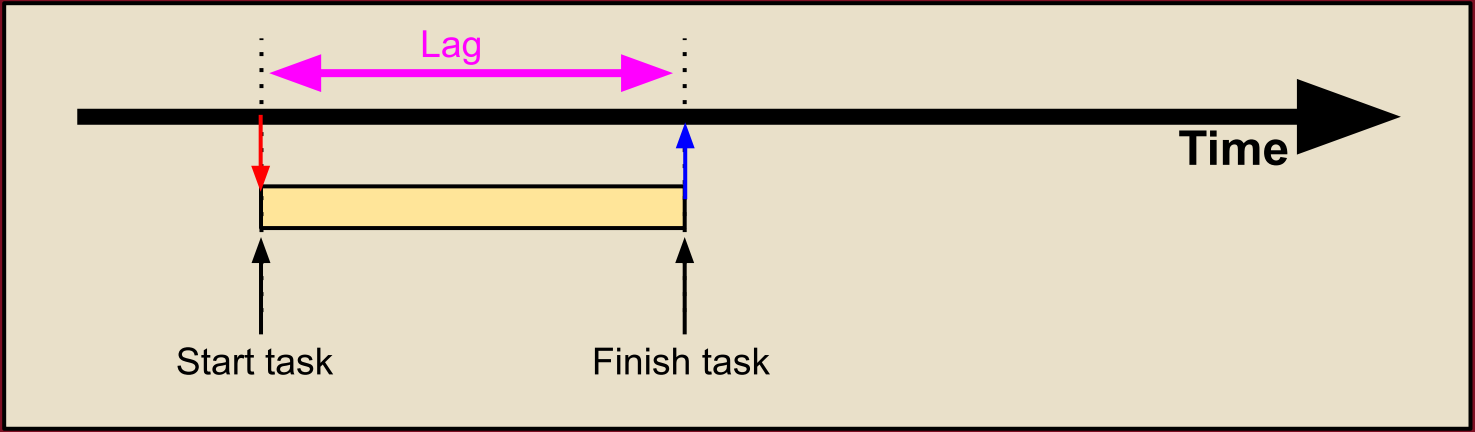
Time choices for Rendering

A choice of times we can choose to render with;

1. Time at *start* of “render”.
2. *Predicted* time at *end* of “render”.

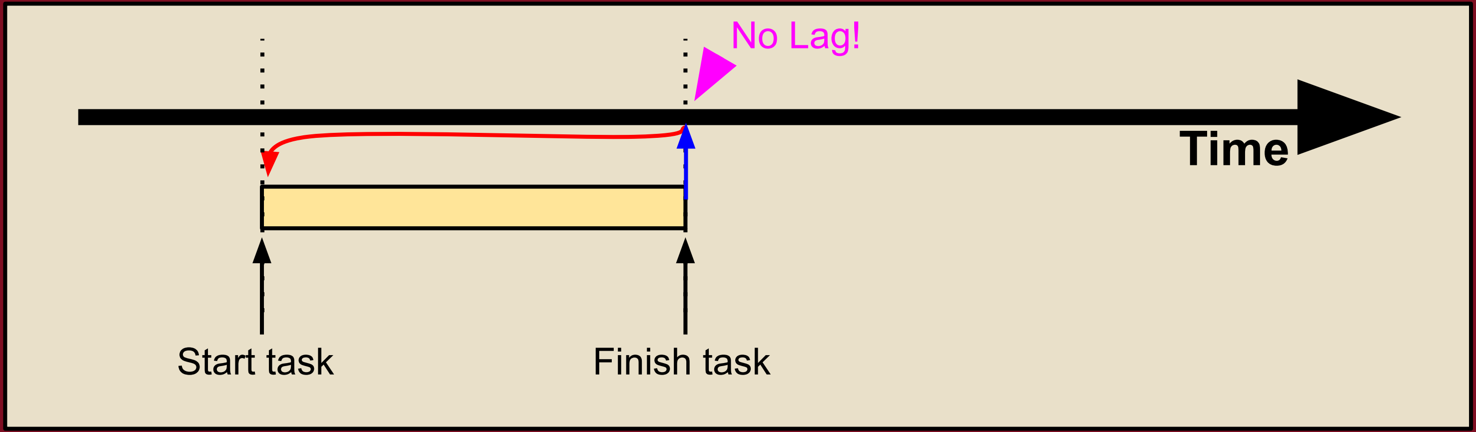
Time at ***start*** of "render"

When you have finished the task,
the world has already moved on!



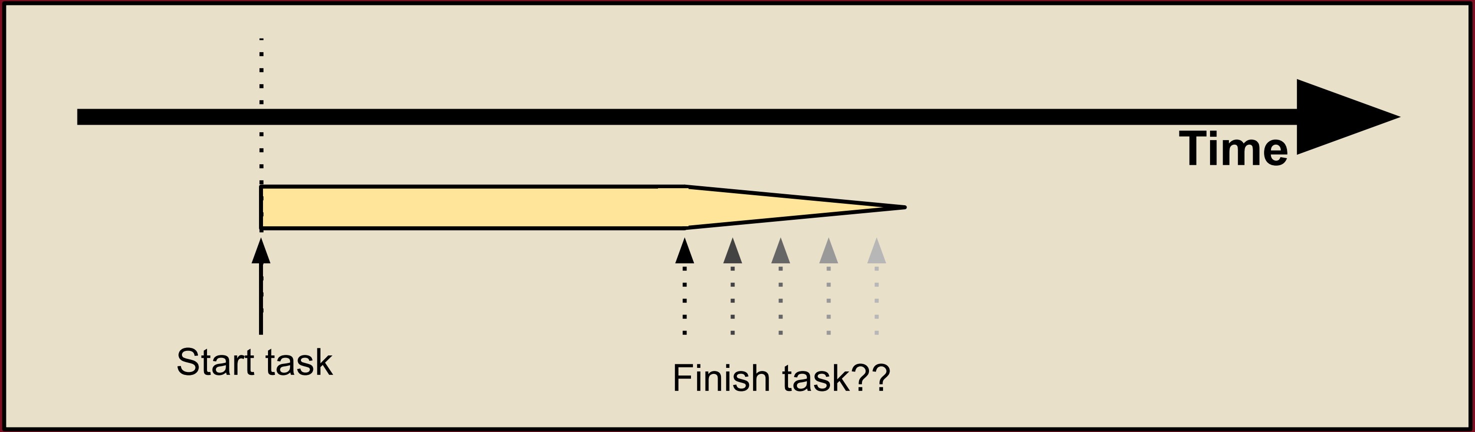
Predicted time at ***end*** of "render"

If we can predict the state, we can use that as the task input and have zero lag.



Need to predicting task length

To bring time backwards you need to be able to predict how long the task will take.



Predicting “Render” Length

In Chrome for arbitrary web content for many parts of the render time it is impossible to know how long they will take.

http://goo.gl/MHGWTc

Glass Time

More complexity.

http://goo.gl/MHGWTc

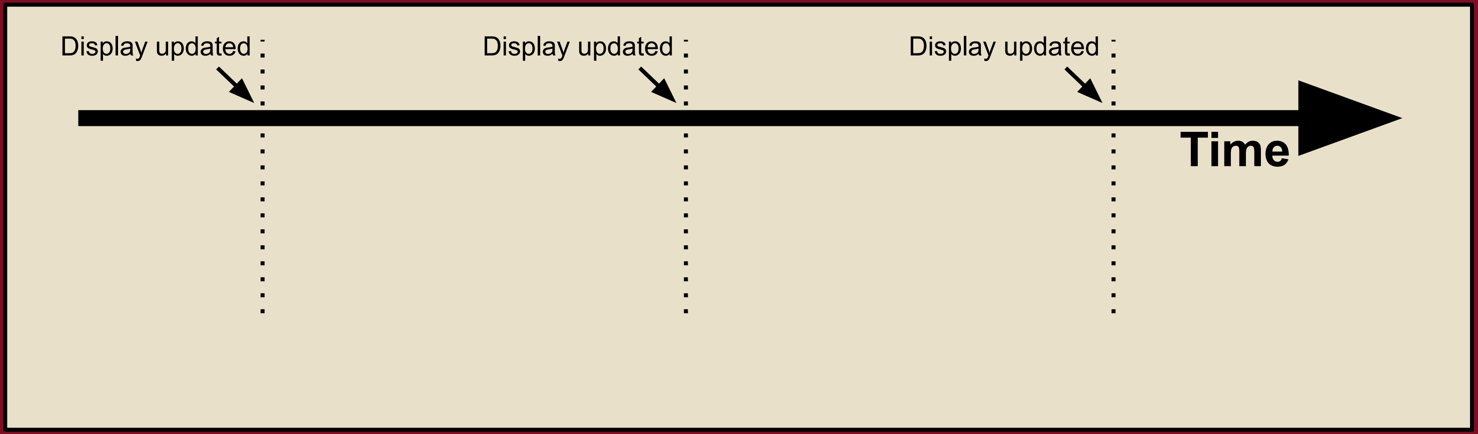
Glass Time

We care about the time something hits the user's eyeball.

This is approximately the same time as the time the output hits the **glass of the display** - hence "**glass time**".

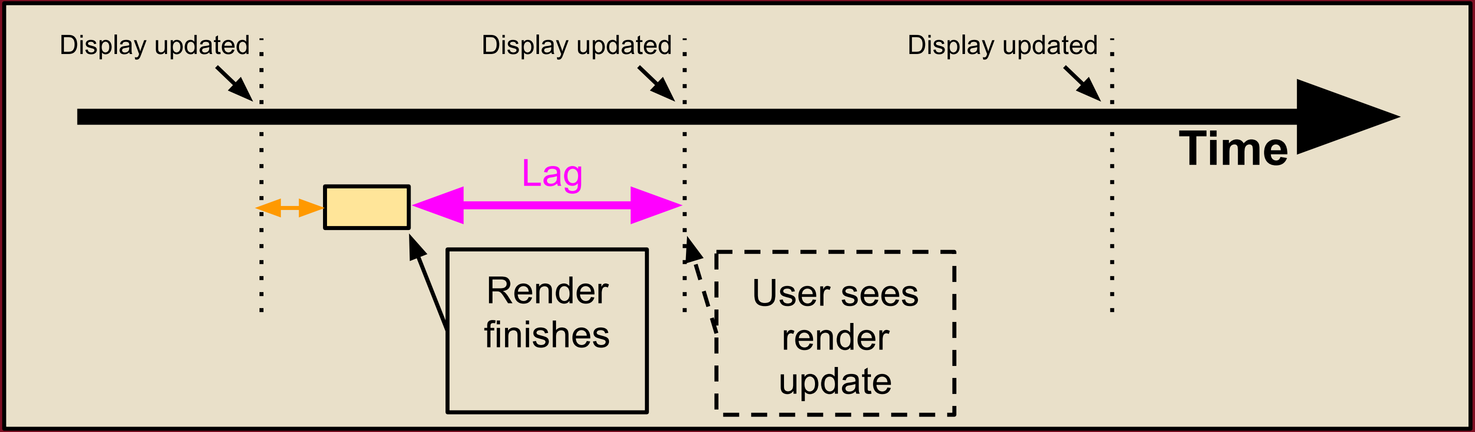
Screen is updated periodically

Current display technologies update periodically.



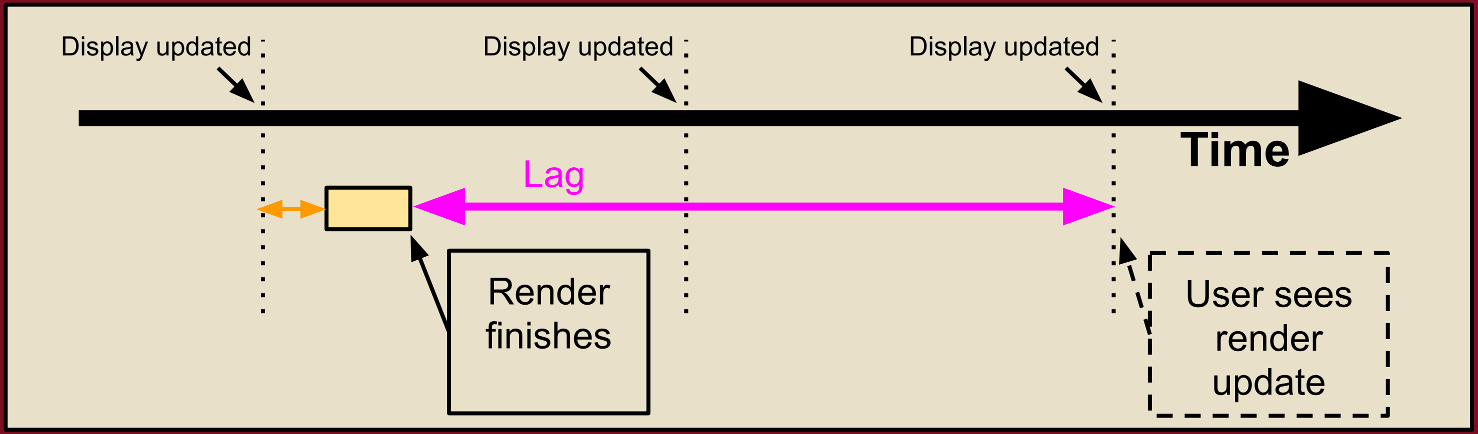
Display lag

Another lag; The lag between finishing "render" and user seeing output.



Display lag

Display lag can be greater than one refresh interval due to buffering/pipelining.



Choice of time used for rendering

Fast render

Assuming **fast** rendering time,

when (render time) < (refresh rate)

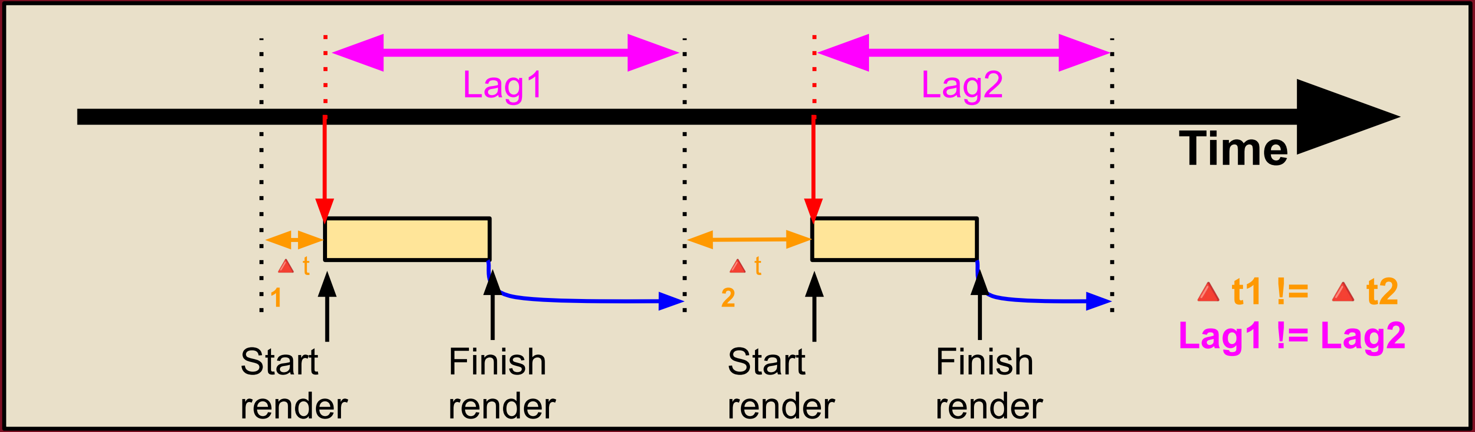
Time Choices for Render

A choice of times we can choose to render with;

1. Time at *start* of “render”.
2. *Predicted* time at *end* of “render”.

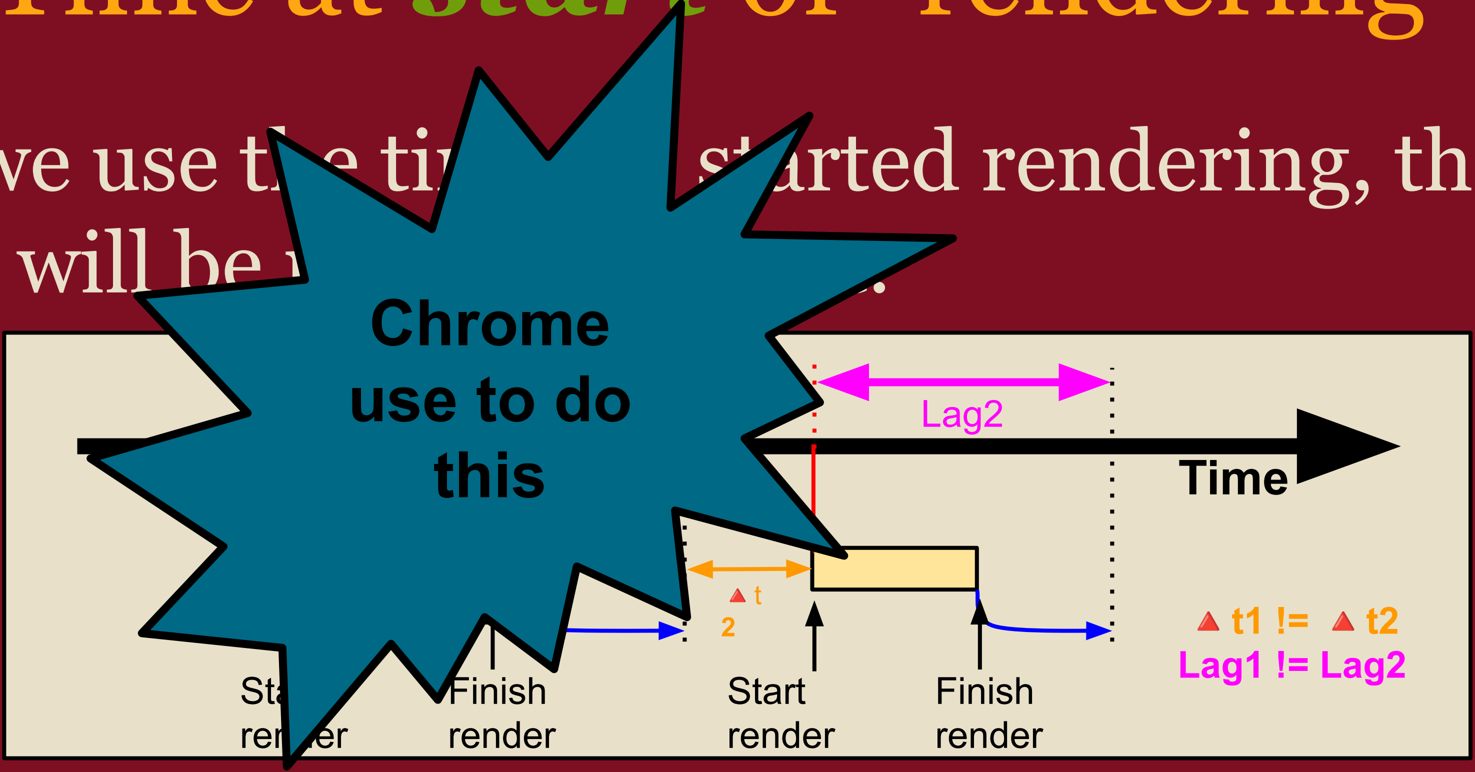
Time at *start* of "rendering"

If we use the time we started rendering, the lag will be **non-constant.**



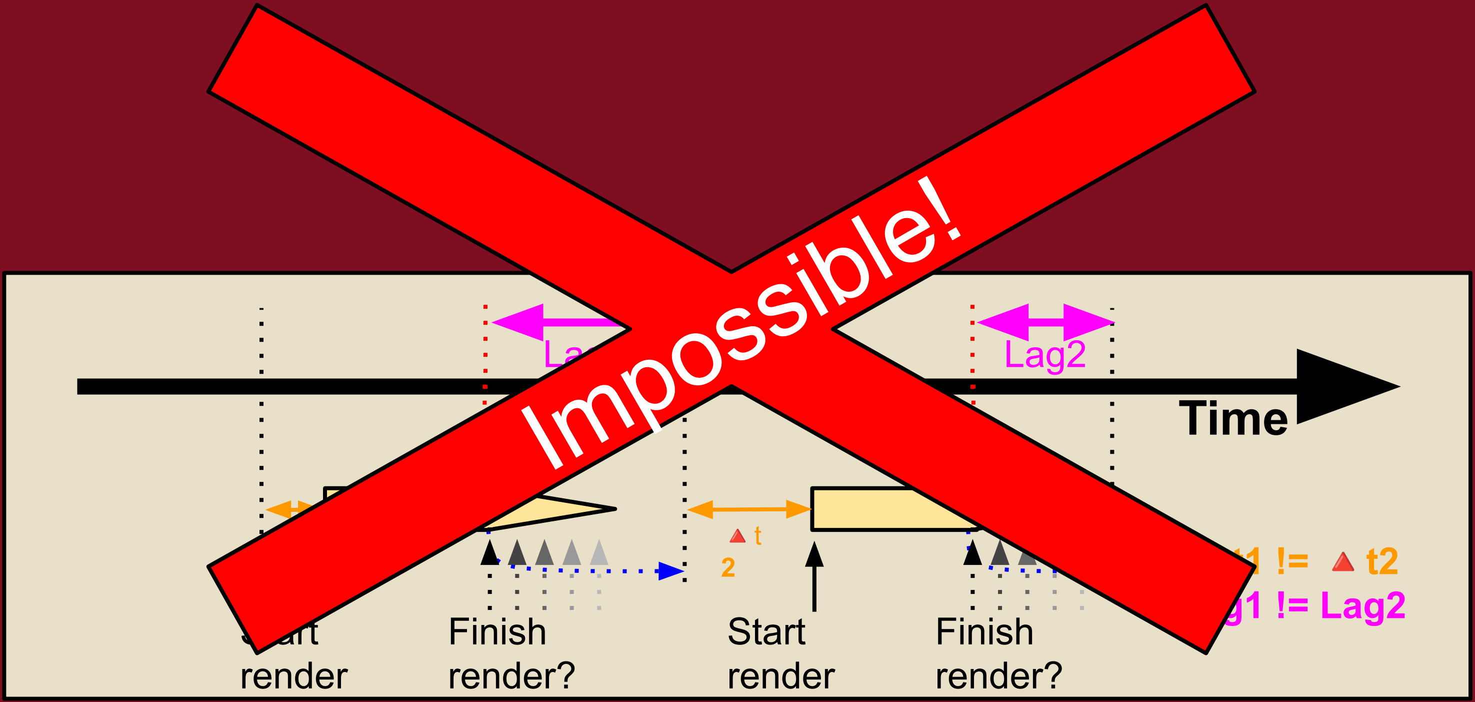
Time at ***start*** of "rendering"

If we use the ti[…] started rendering, the lag will be […].



Fast render time

Predicted time at ***end*** of "render"



Fast render time

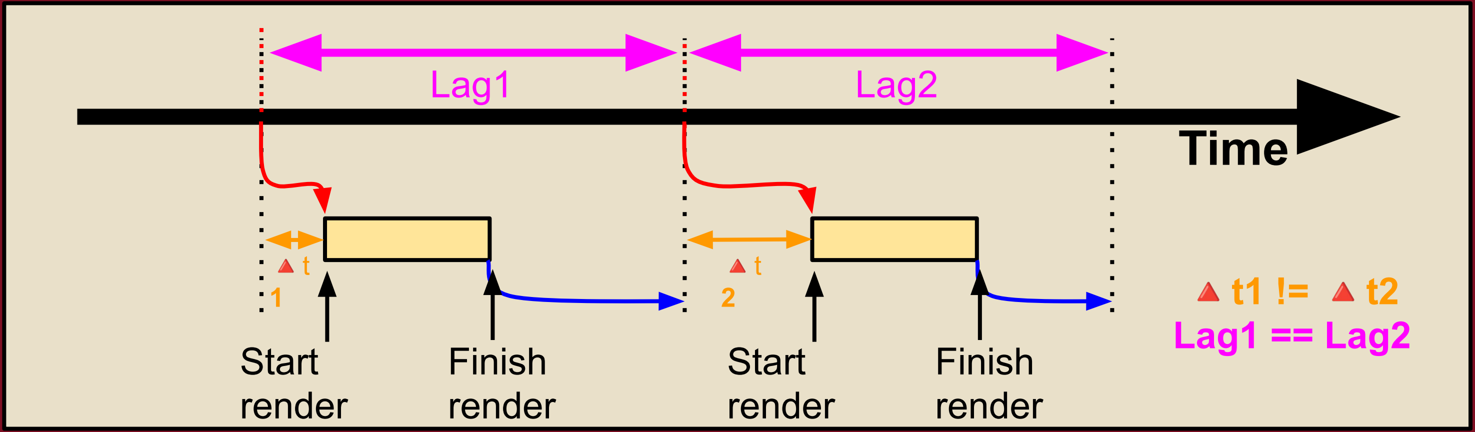
Time Choices for Render

A choice of times we can choose to render with;

1. Time at *start* of "render".
2. *Predicted* time at *end* of "render".
3. Last **glass time** *before* "render".

Last glass time **before** "render"

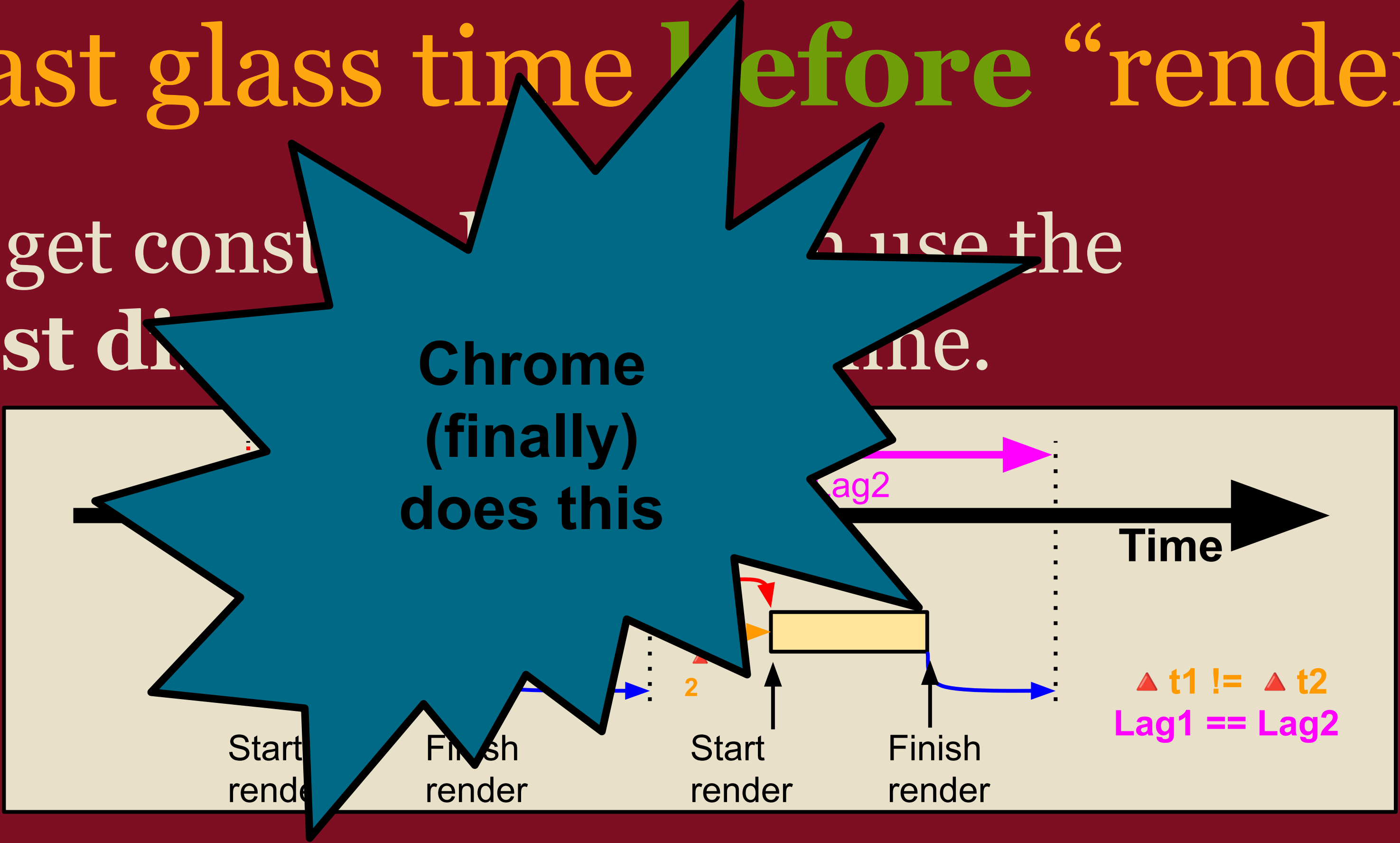
To get constant lag, we can use the "**last display updated**" time.



Fast render time

Last glass time before "render"

To get const... use the "last di... ...me.



Fast render time

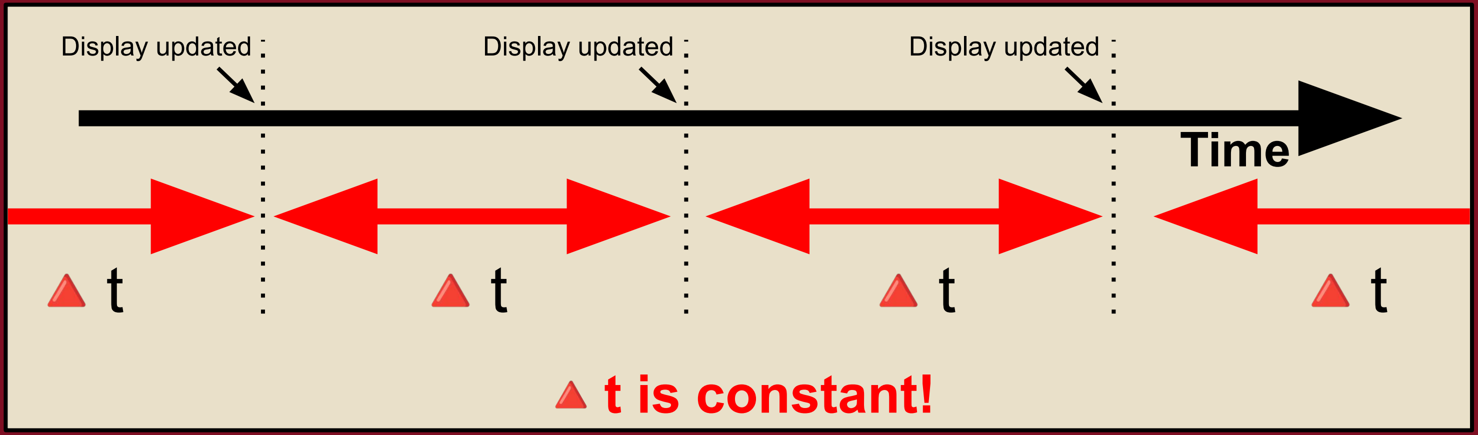
Time Choices for Render

A choice of times we can choose to render with;

1. Time at *start* of “render”.
2. *Predicted* time at *end* of “render”.
3. Last **glass time** *before* “render”.
4. *Predicted* **glass time** *after* “render”.

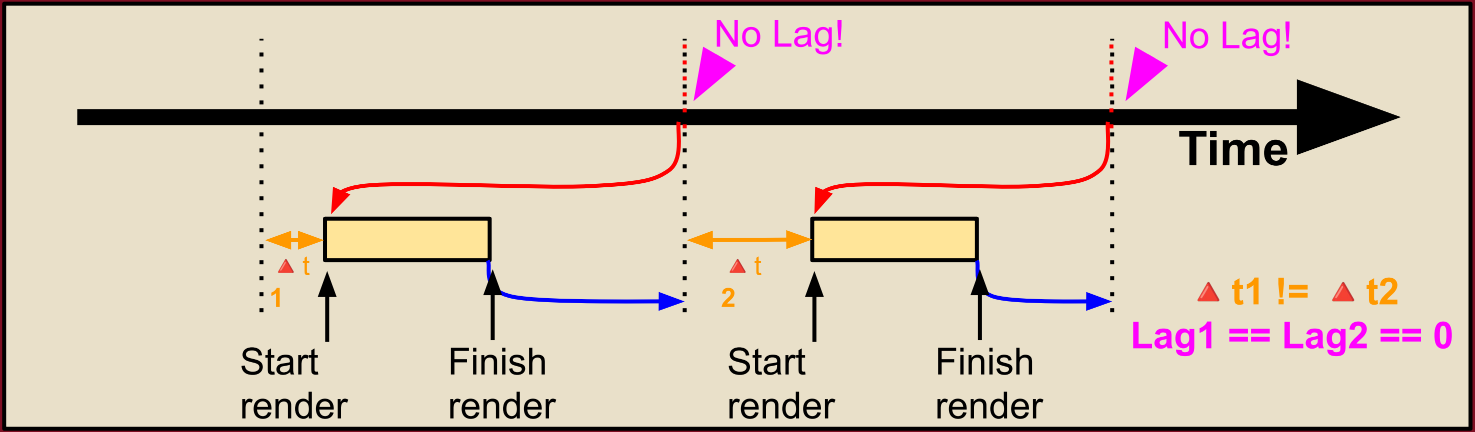
Predicting "glass time"

The update time is **stable and predictable.**



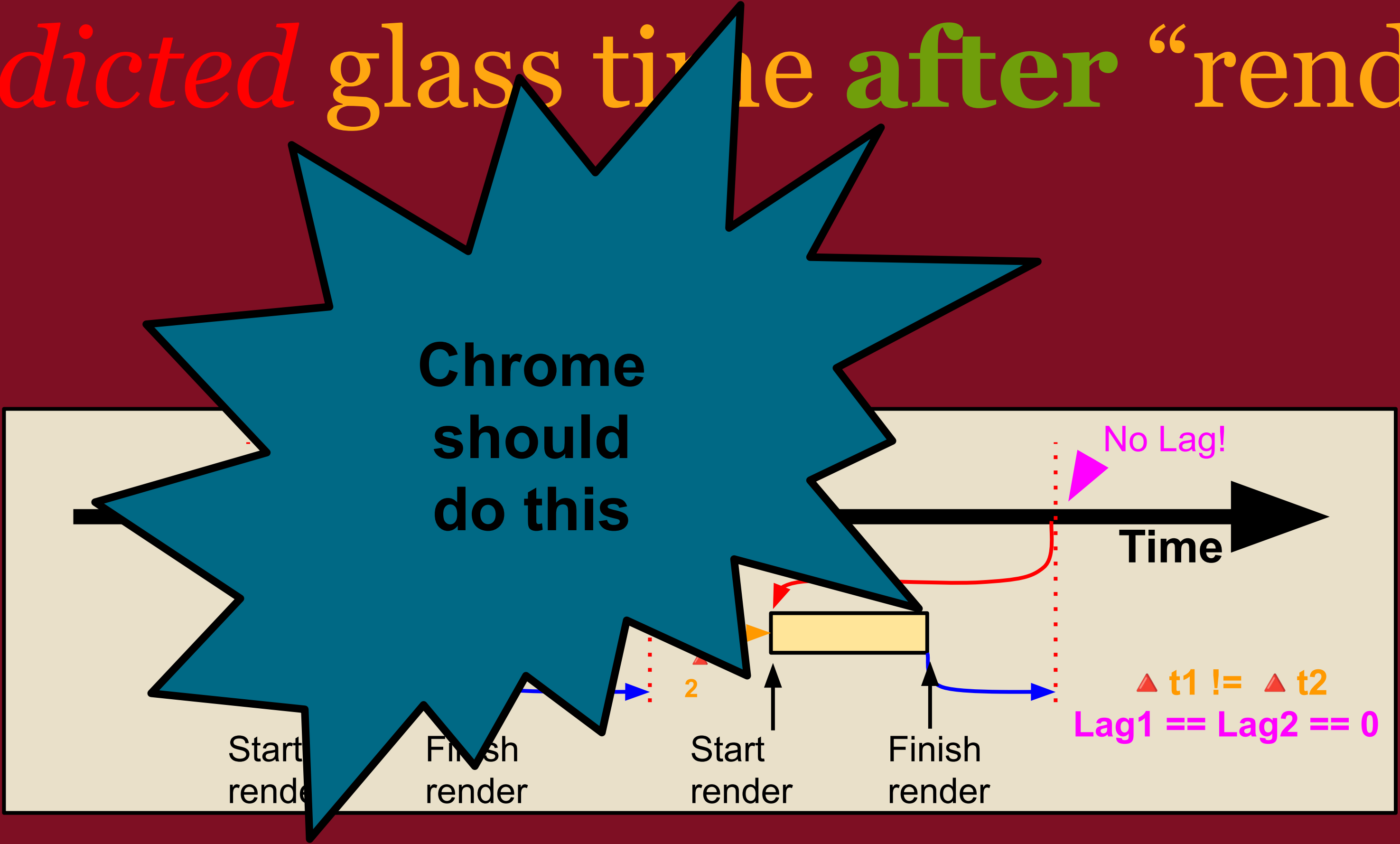
Fast render time

Predicted glass time **after** “render”



Fast render time

Predicted glass time **after** "render"



Fast render time

Summary

~~[Bad]~~ ~~Time at *start* of "render".~~

~~[Bad]~~ ~~*Predicted* time at *end* of "render".~~

[Okay] Last **glass time** *before* "render".

[Best] *Predicted* **glass time** *after* "render".

Summary for Chrome

[Past] Time at *start* of “render”.

~~**[Impossible]** *Predicted* time at *end* of “render”.~~

[Now] Last **glass time** *before* “render”.

[Future] *Predicted* **glass time** *after* “render”.

Fast render time

http://goo.gl/MHGWTc

Choice of time used for rendering

Slow render

Back to Back rendering

Assuming **slow** rendering time,

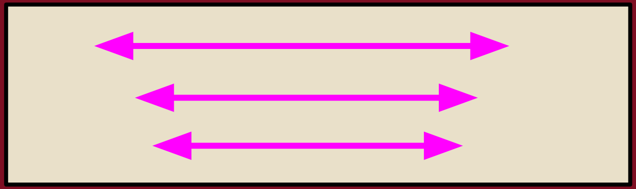
when (render time) > (refresh period)

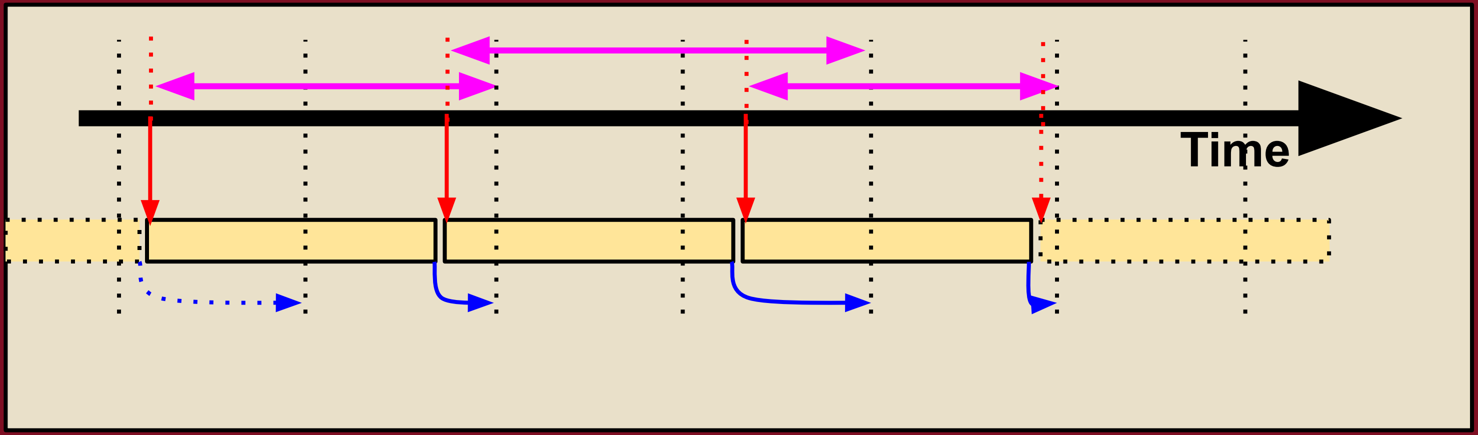
Time Choices for Render

A choice of times we can choose to render with;

1. Time at *start* of "render".
2. *Predicted* time at *end* of "render".
3. Last **glass time** *before* "render".
4. *Predicted* **glass time** *after* "render".

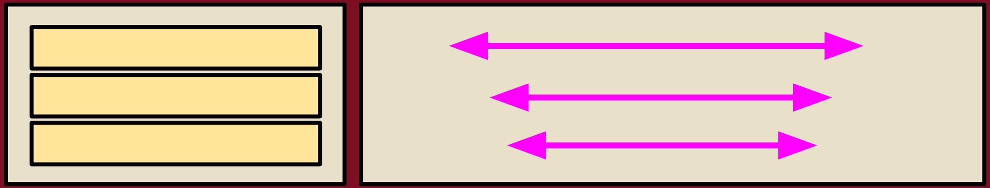
Time at ***start*** of "rendering"

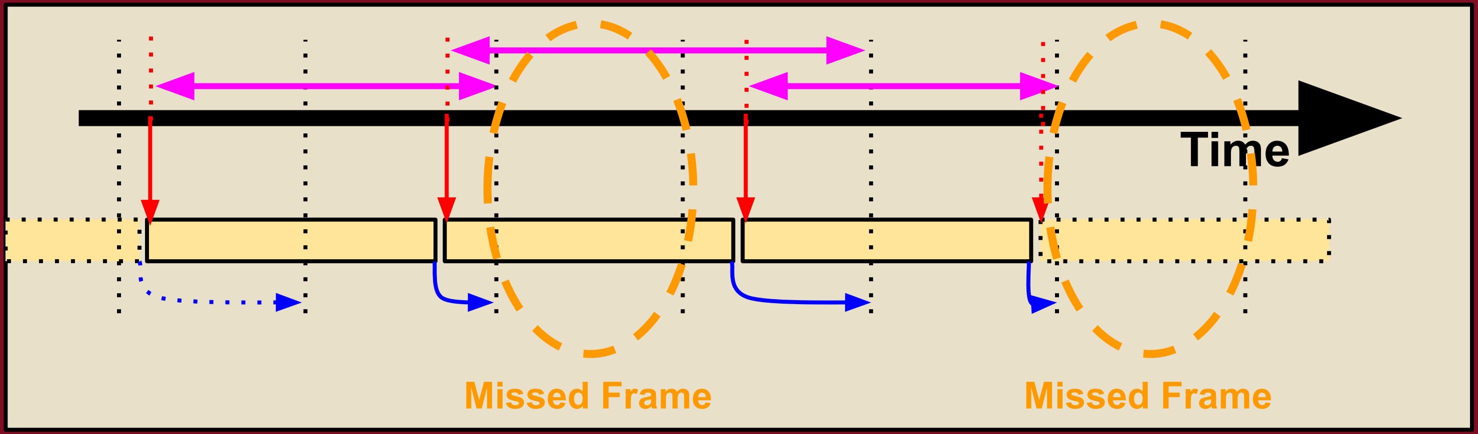




Slow render time - Back to Back

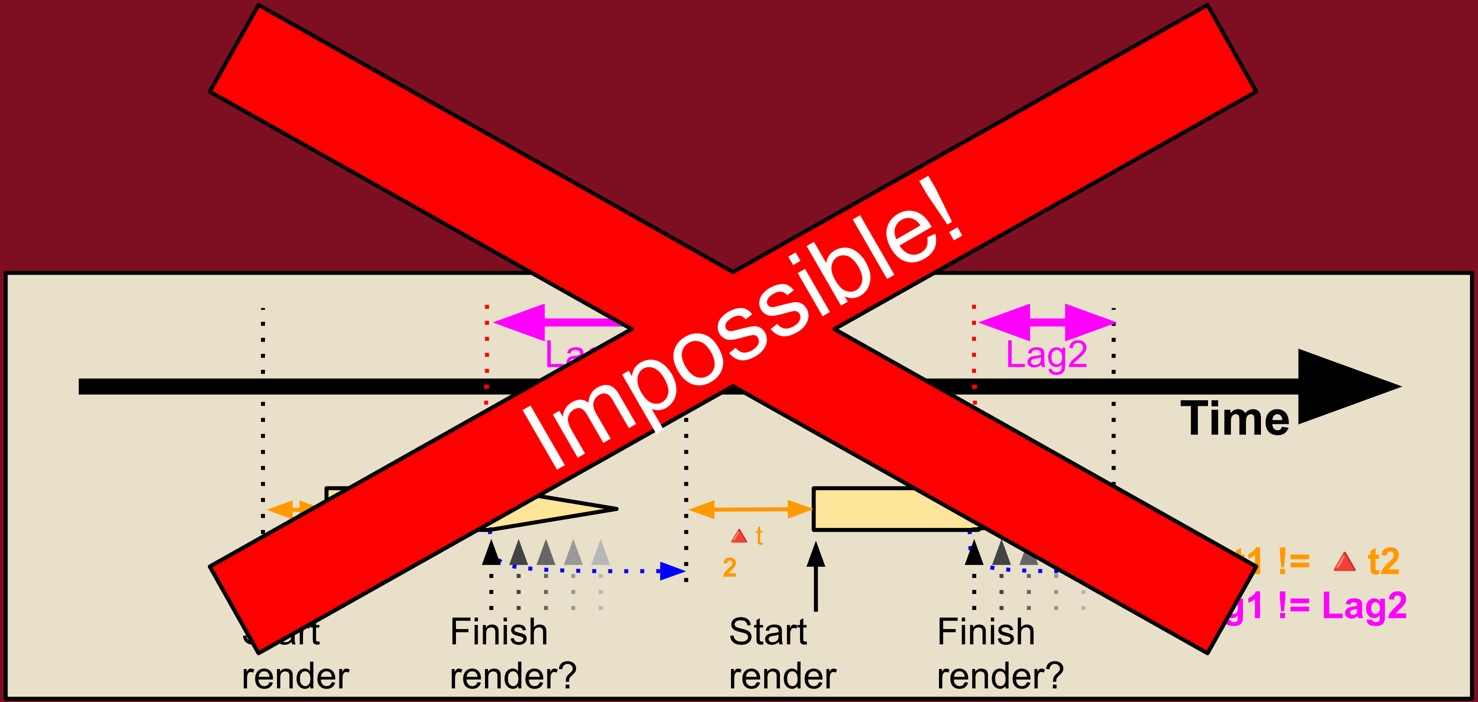
Time at *start* of "rendering"





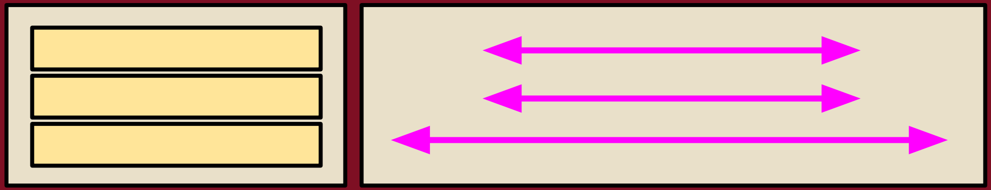
Slow render time - Back to Back

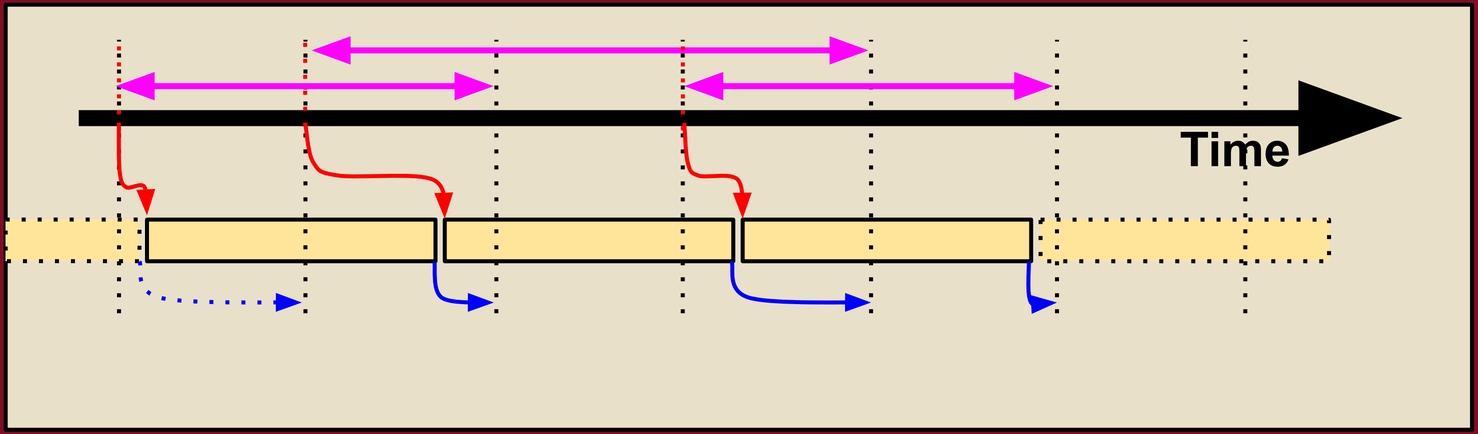
Predicted time at ***end*** of "render"



Slow render time - Back to Back

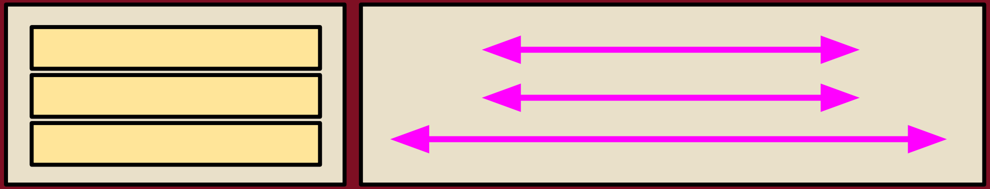
Last glass time **before** “render”

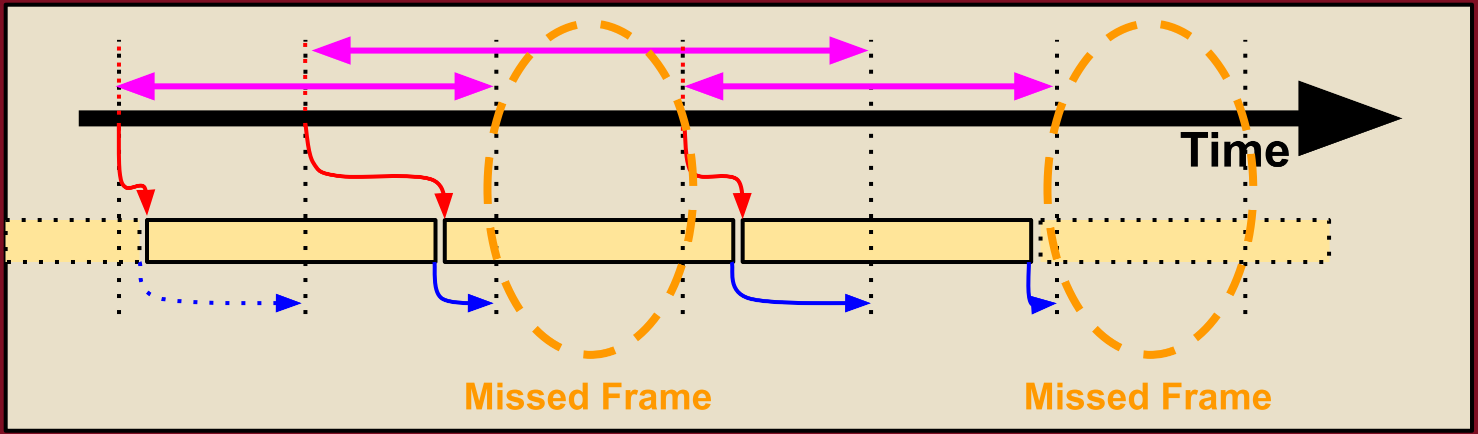




Slow render time - Back to Back

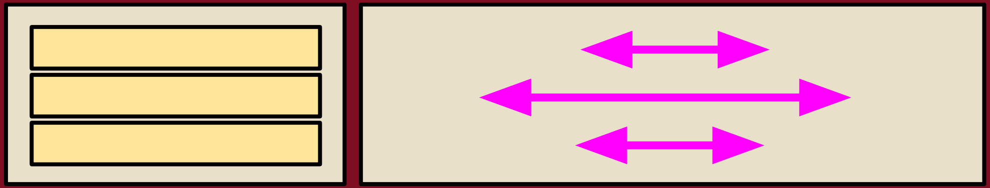
Last glass time **before** “render”

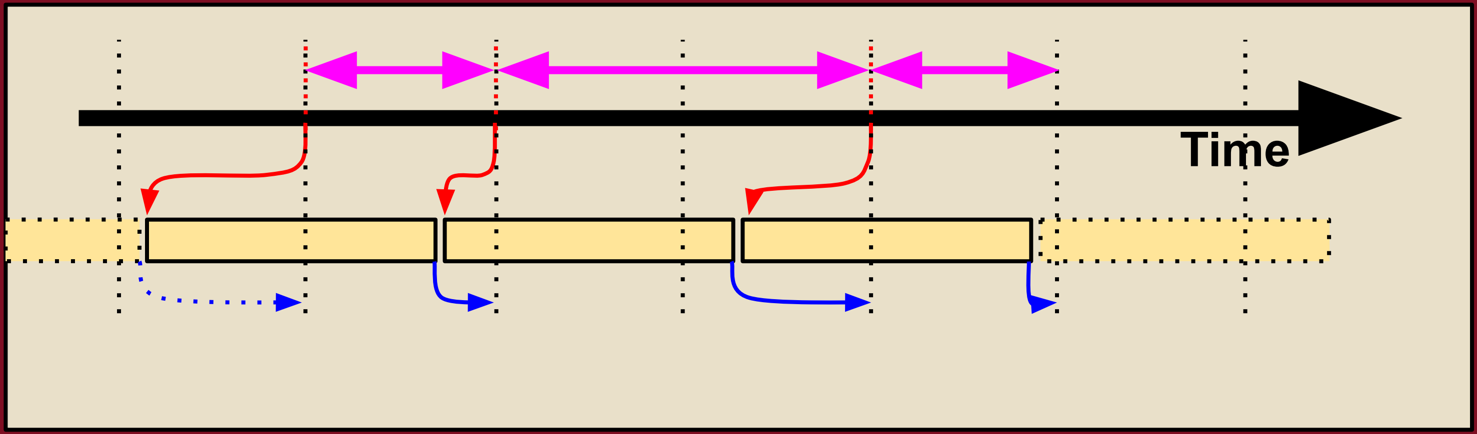




Slow render time - Back to Back

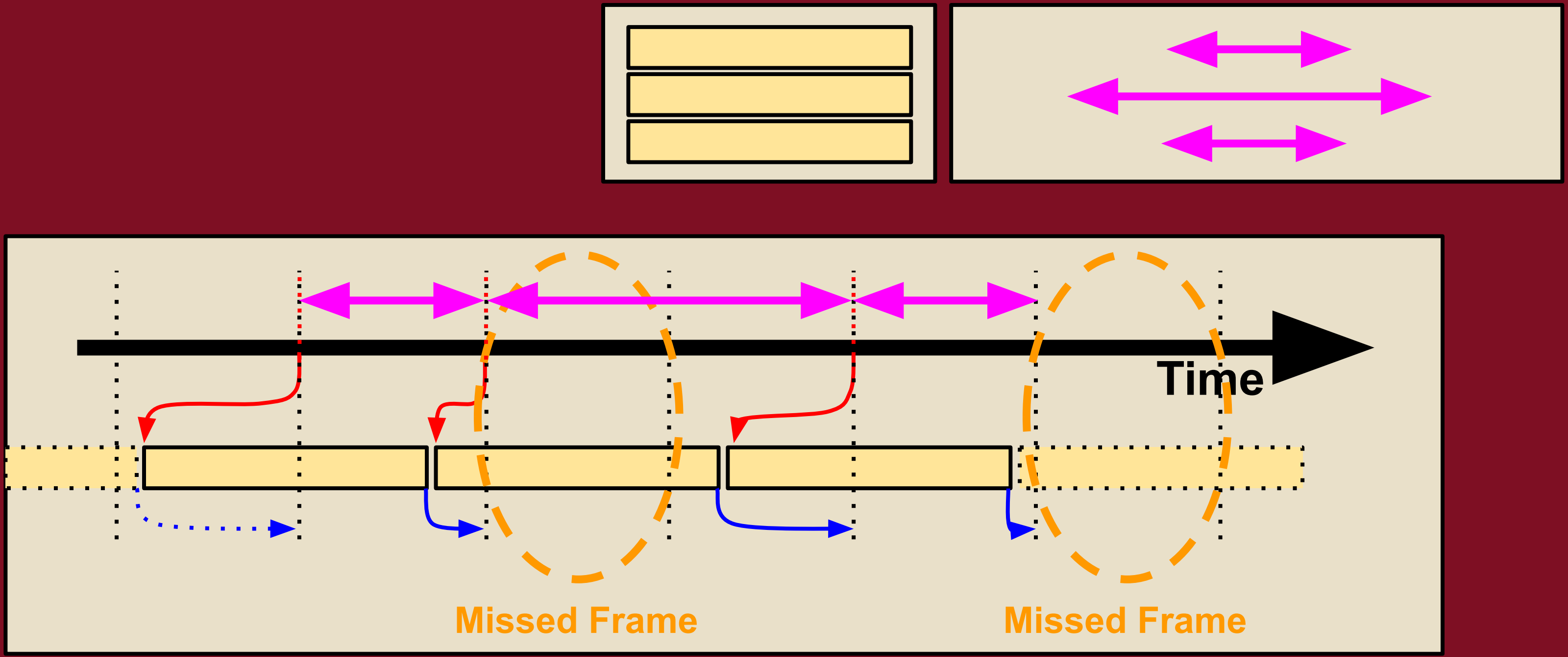
Predicted glass time **after** “render”





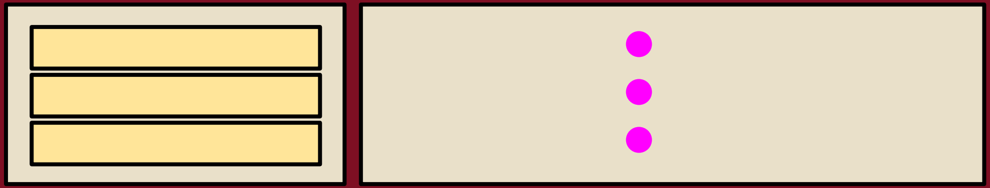
Slow render time - Back to Back

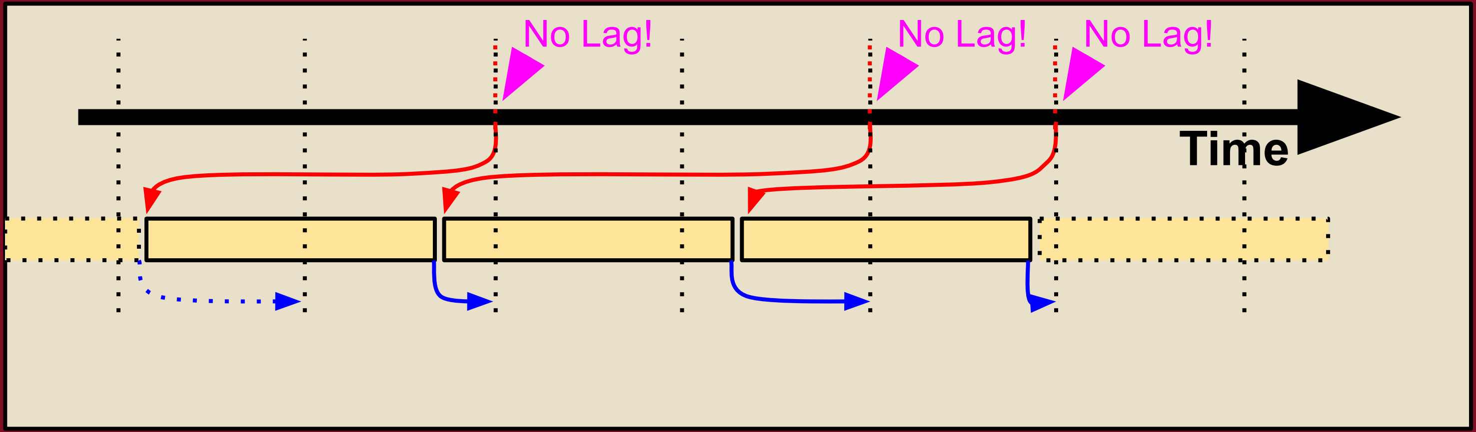
Predicted glass time **after** “render”



Slow render time - Back to Back

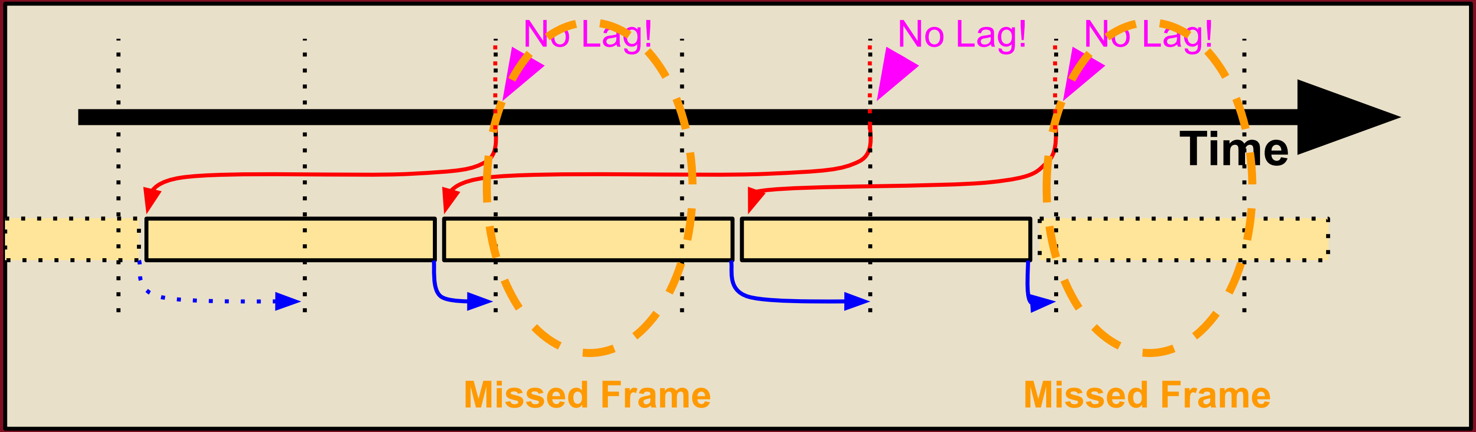
Predicted glass time **after** "render"





Slow render time - Back to Back

Predicted glass time **after** “render”



Slow render time - Back to Back

Summary

~~**[Bad]**~~ ~~Time at *start* of “render”.~~

~~**[Bad]**~~ ~~*Predicted* time at *end* of “render”.~~

~~**[Bad]**~~ ~~Last **glass time** *before* “render”.~~

[Kinda] *Predicted* **glass time** *after* “render”.

Summary for Chrome

Back to Back mode rarely used, only occurs when “frame throttling” is disabled.

All platforms enabled “frame throttling” by default.

http://goo.gl/MHGWTc

Choice of time used for rendering

Slow render time

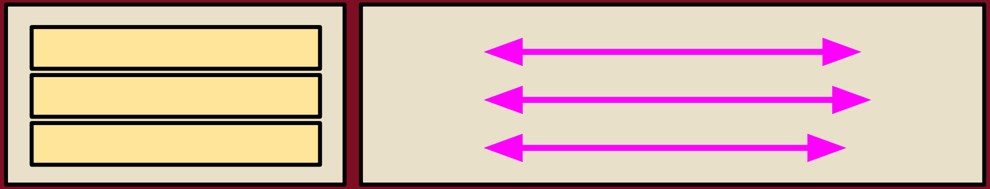
Frame Skip Rendering

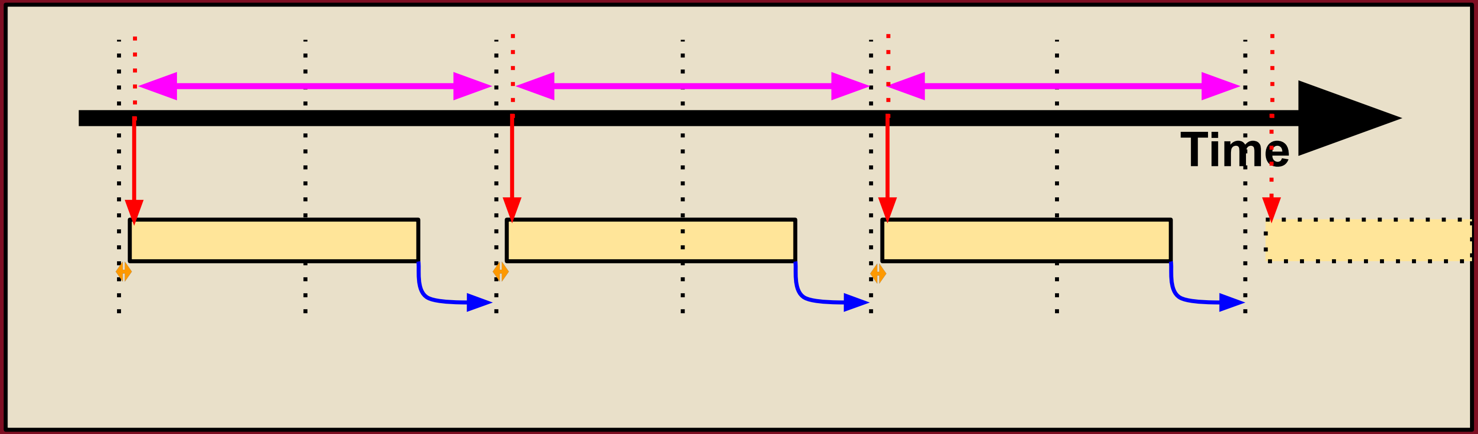
Assuming **slow** render time,

when (render time) > (refresh period)

http://goo.gl/MHGWTc

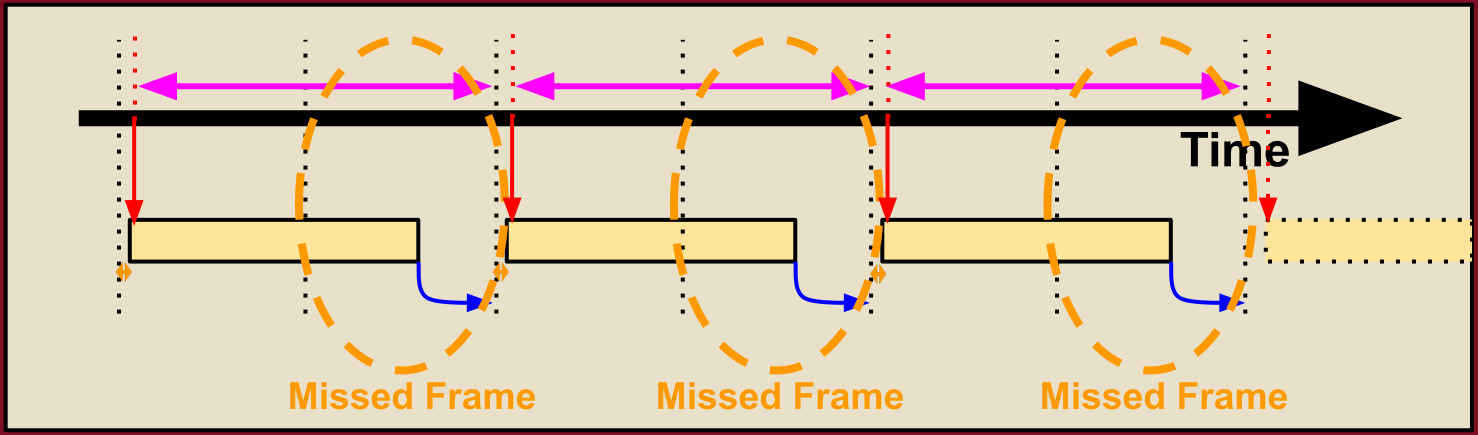
Time at ***start*** of “rendering”





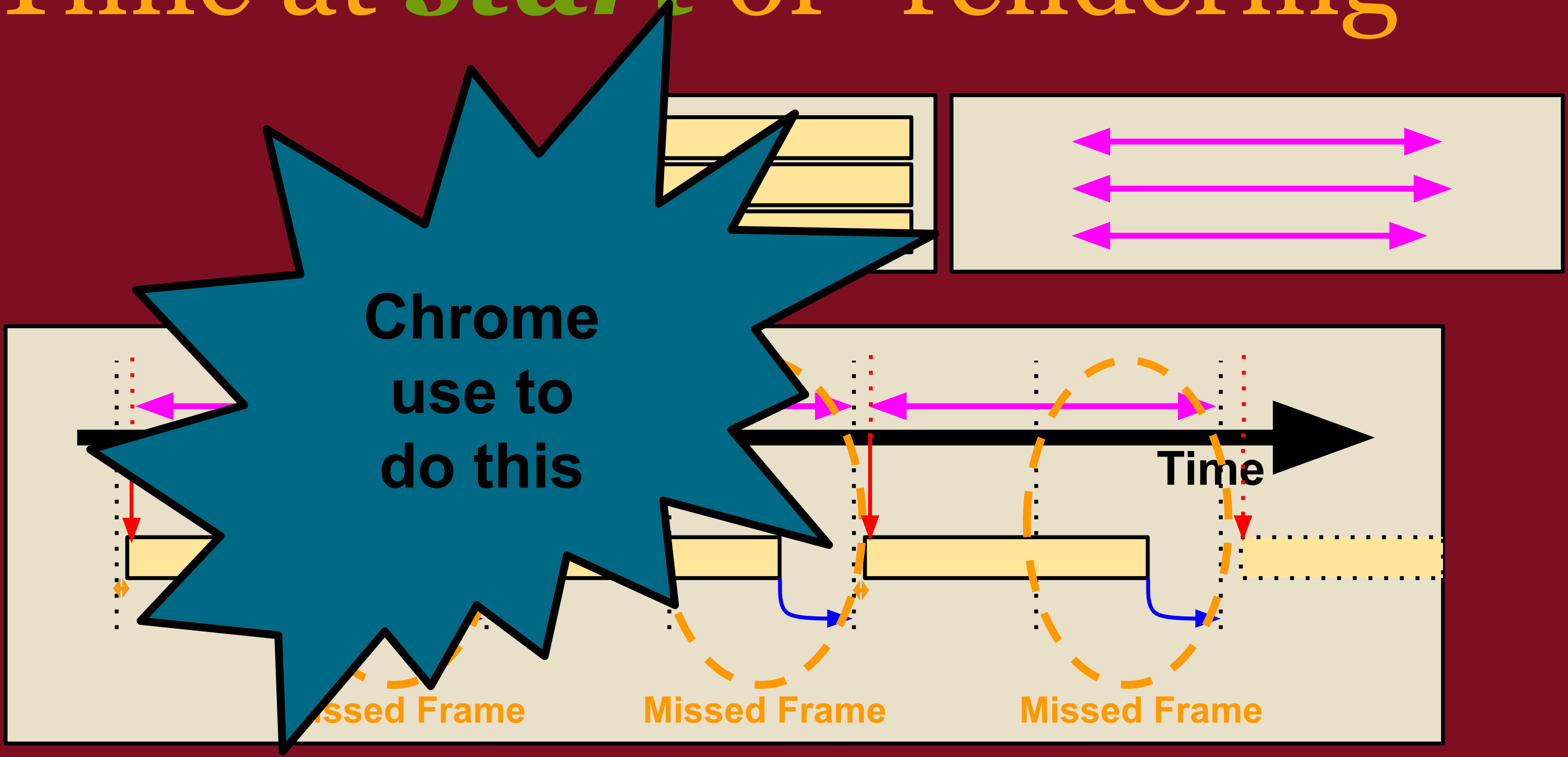
Slow render time - Frame Skipping

Time at ***start*** of "rendering"



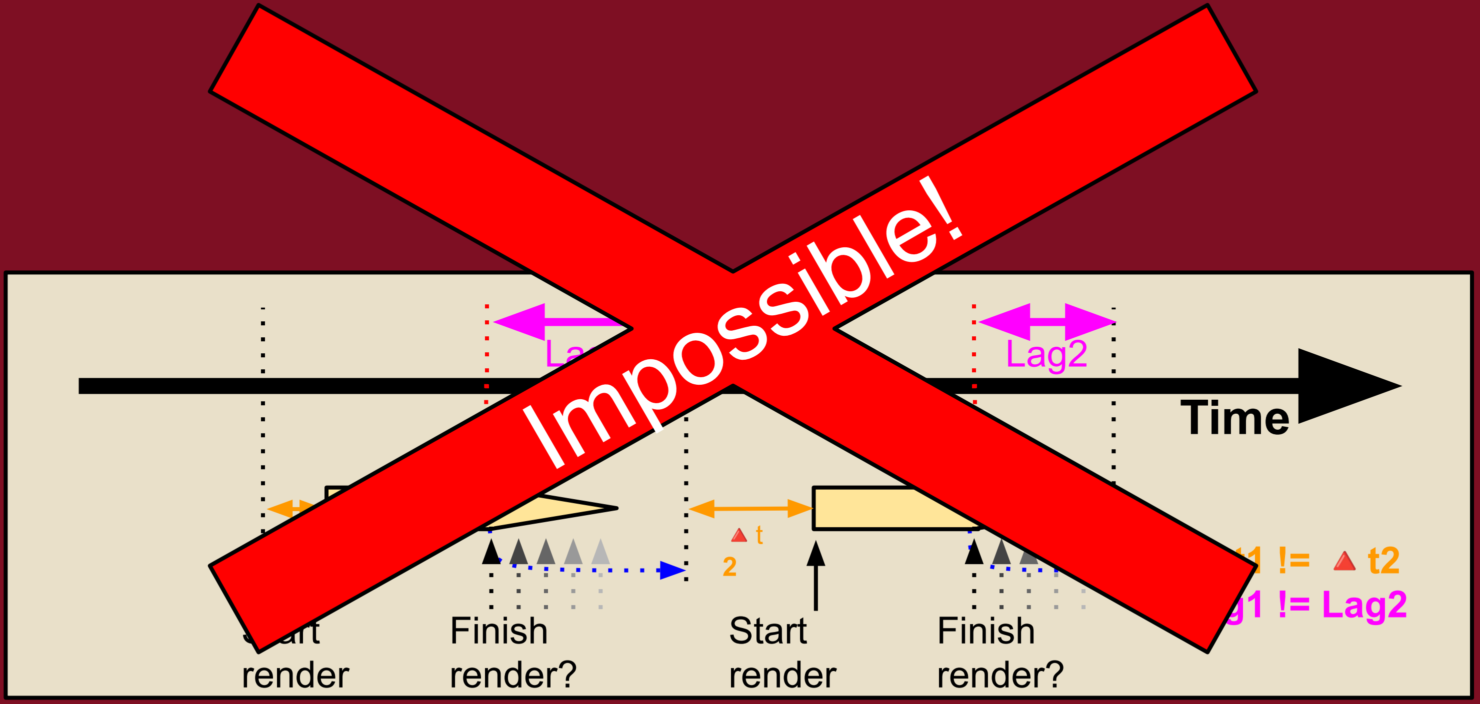
Slow render time - Frame Skipping

Time at *start* of "rendering"



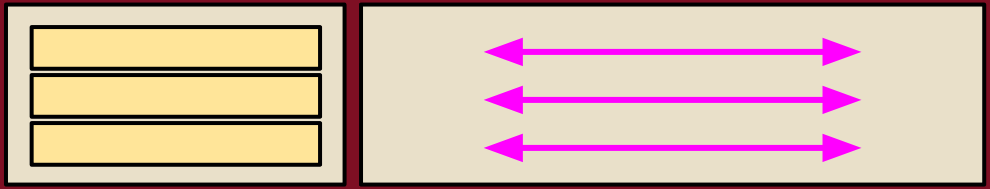
Slow render time - Frame Skipping

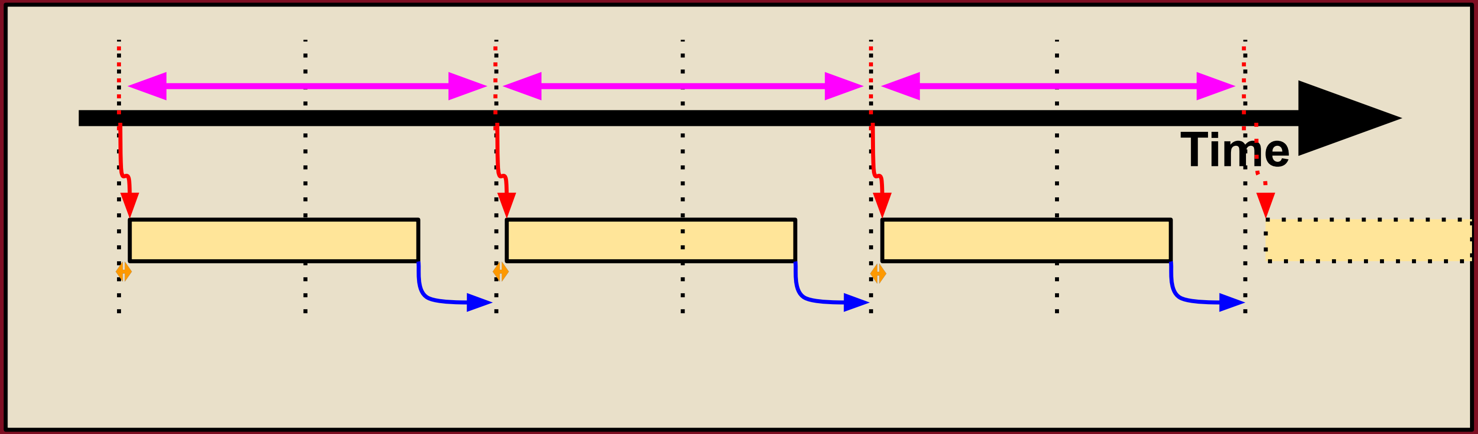
Predicted time at ***end*** of “render”



Slow render time - Back to Back

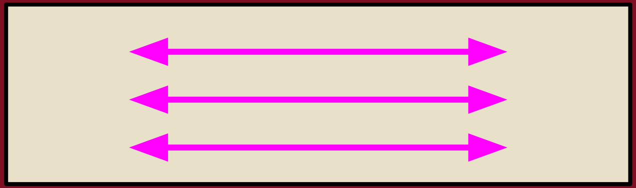
Last glass time **before** “render”

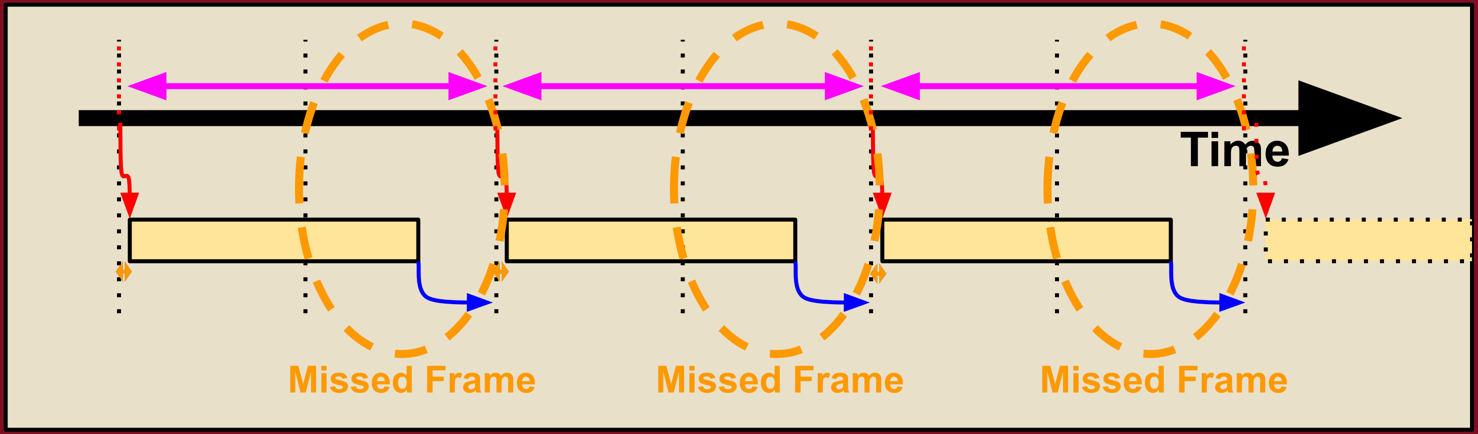




Slow render time - Frame Skipping

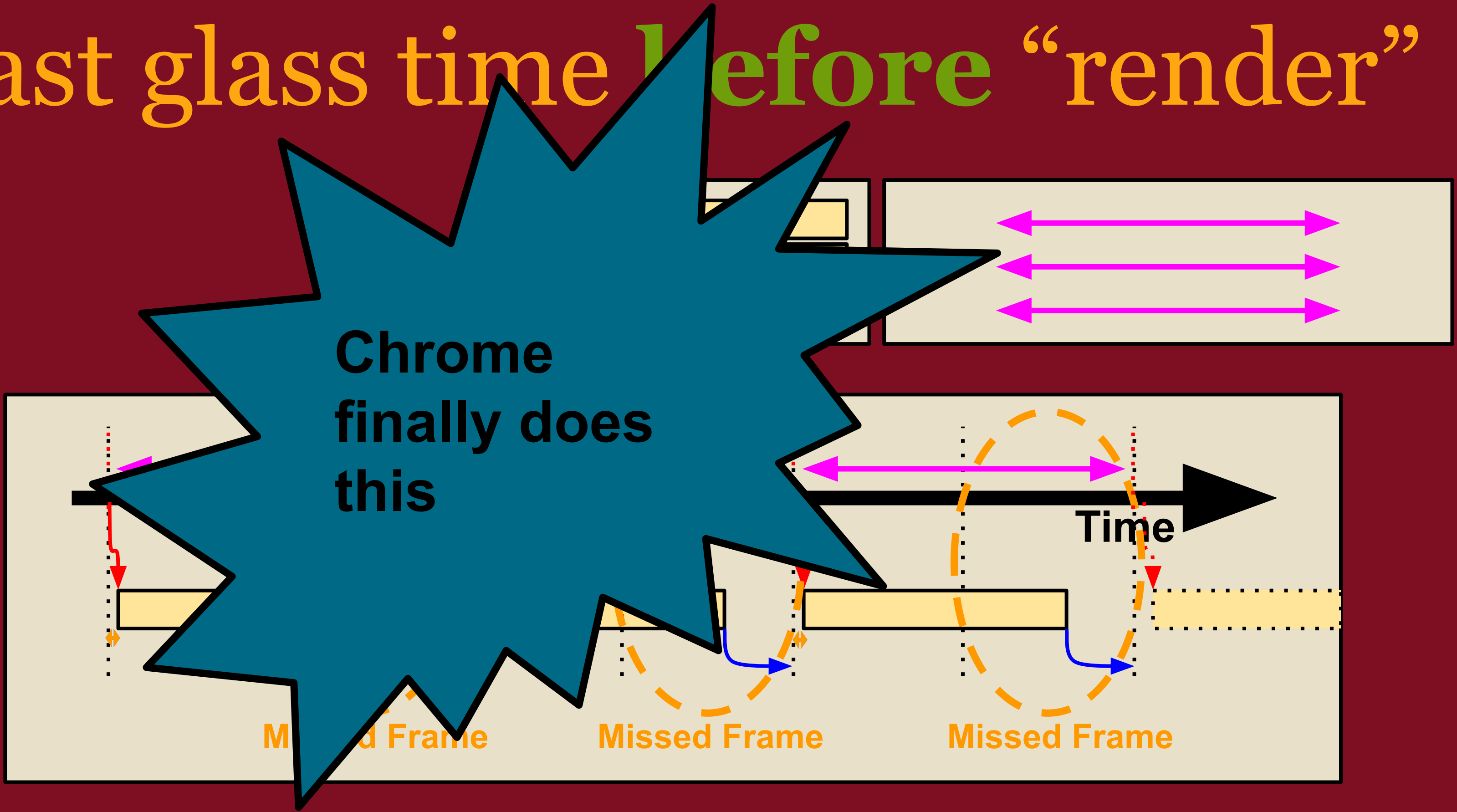
Last glass time **before** “render”





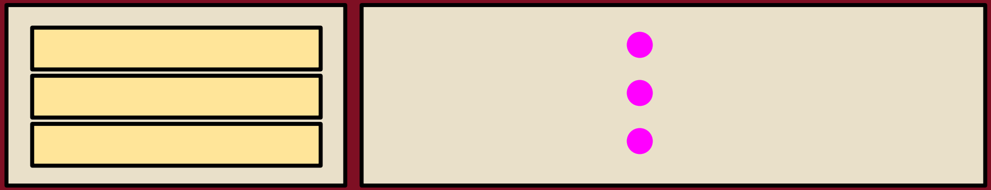
Slow render time - Frame Skipping

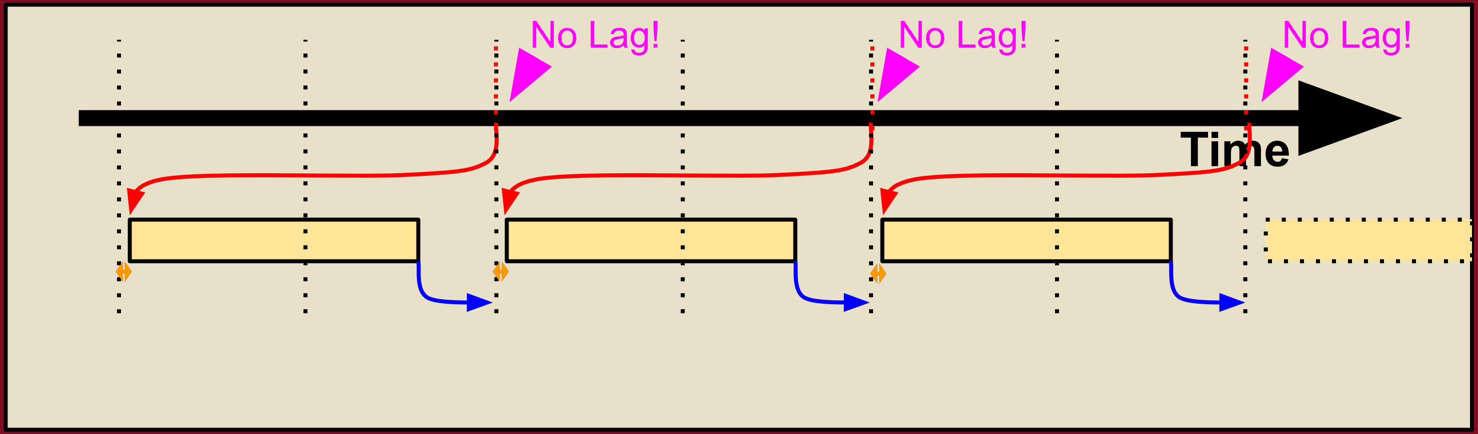
Last glass time **before** "render"



Slow render time - Frame Skipping

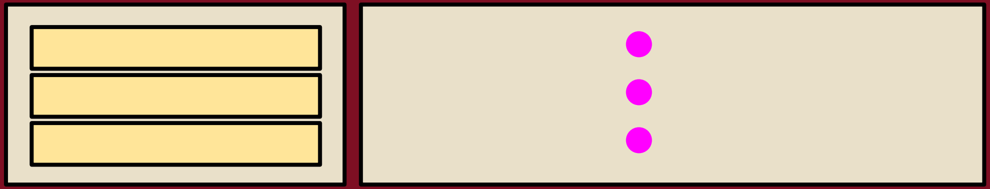
Predicted glass time **after** “render”

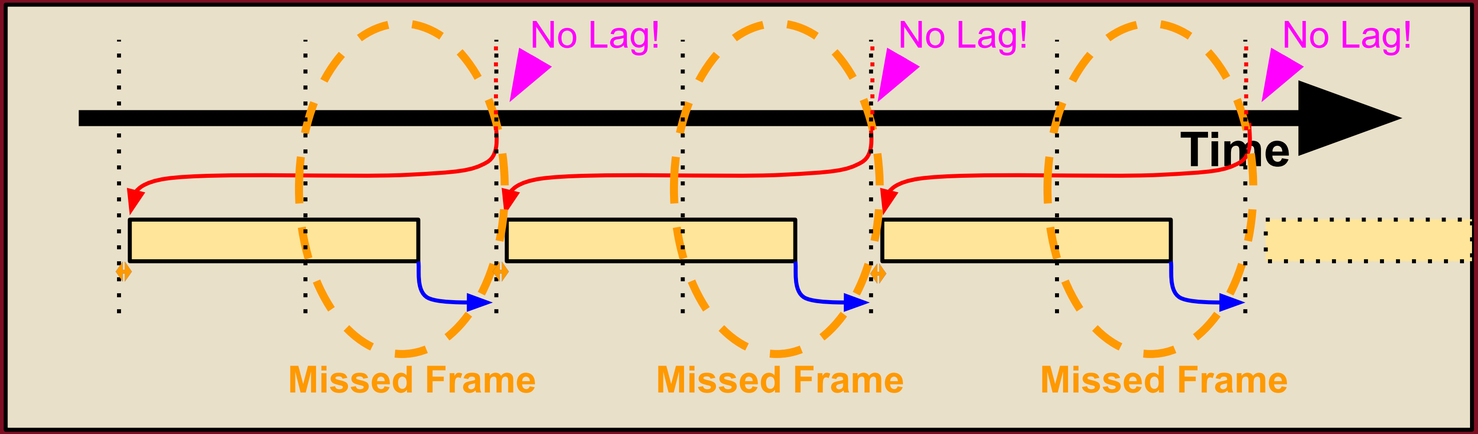




Slow render time - Frame Skipping

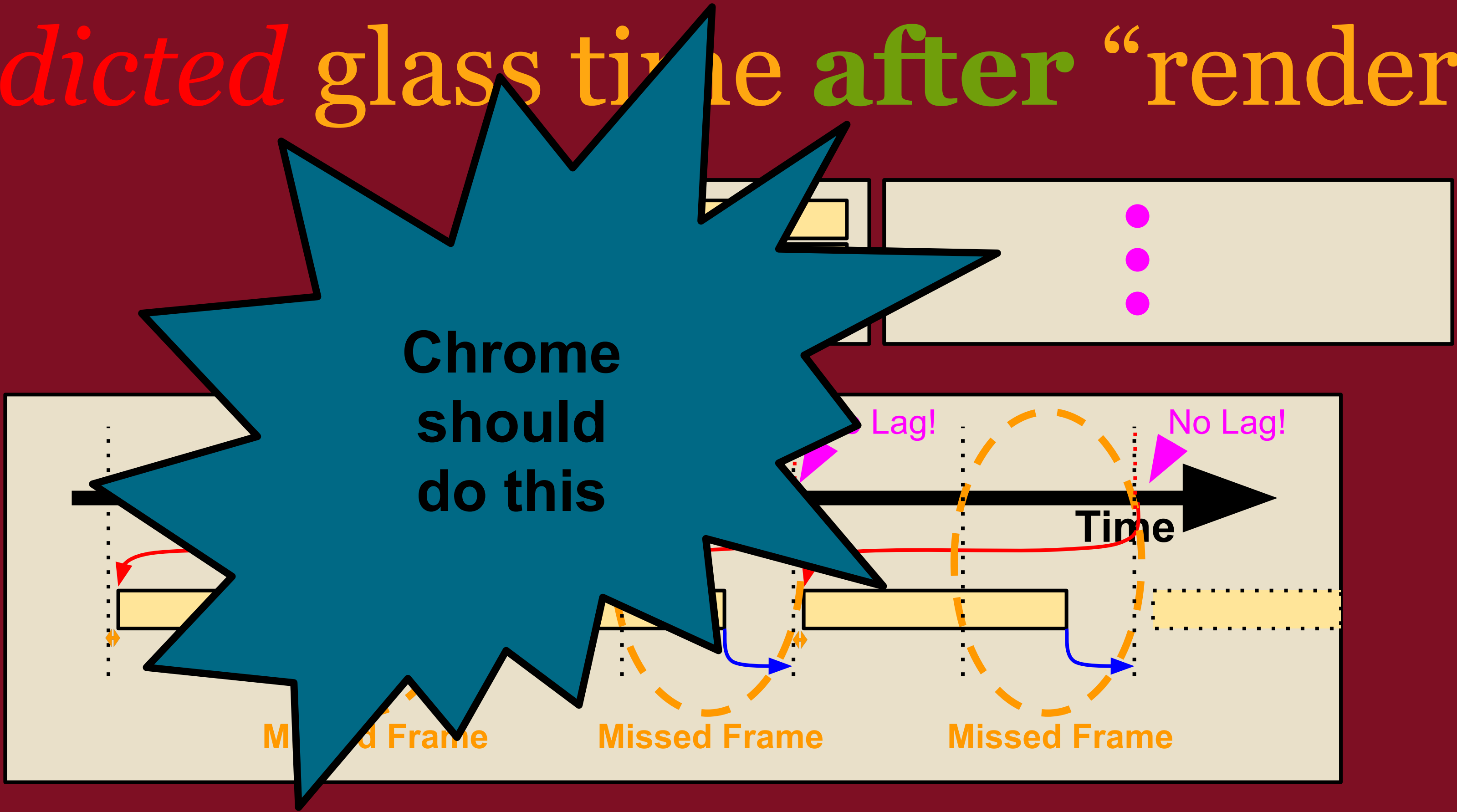
Predicted glass time **after** "render"





Slow render time - Frame Skipping

Predicted glass time **after** “render”



Slow render time - Frame Skipping

Summary

~~[Bad] Time at *start* of "render".~~

~~[Bad] *Predicted* time at *end* of "render".~~

[Okay] Last **glass time** *before* "render".

[Best] *Predicted* **glass time** *after* "render".

Summary for Chrome

[Current] Time at *start* of "render".

~~**[Impossible]** *Predicted* time at *end* of "render".~~

[Soon] Last **glass time** *before* "render".

[Future] *Predicted* **glass time** *after* "render".

Slow render time - Frame Skipping

http://goo.gl/MHGWTc

Time Choices for Render

A choice of times we can choose to render with;

- Time at *start* of “render”.
- *Predicted* time at *end* of “render”.
- Last **glass time** *before* “render”.
- *Predicted* **glass time** *after* “render”.

Time at *start* of "rendering"

Fast Rendering

- **Variable Lag**

Slow Rendering - Back to Back

- **Variable Lag**

Slow Rendering - Frame Skipping

- **Variable Lag**

Conclusion: Always bad!

Predicted time at ***end*** of "render"

Fast Rend… g

- **Impossible**

Slow Rendering - … Back

- **Impossible**

Slow Renderi… Frame S… ing

- **Imposs… e**

Conclusion: Impossible!

Last glass time **before** “render”

Fast Rendering

- **Constant Lag**

Slow Rendering - Back to Back

- **Variable Lag**

Slow Rendering - Frame Skipping

- **Constant Lag**

Conclusion: Mostly okay.

Predicted glass time **after** "render"

Fast Rendering

- **Zero Lag**

Slow Rendering - Back to Back

- **Variable Lag, Fixable to Zero Lag**

Slow Rendering - Frame Skipping

- **Variable Lag, Very fixable to Zero Lag**

Conclusion: Lets do this!

Process forward for Chrome / Blink

http://goo.gl/MHGWTc

Process forward for Chrome

1. Make everything used passed in frame time. Then at;
   Last **glass time** *before* “render”.
2. Make vsync source stable / add glass time predictor.
3. Pass in “glass time”. Then at;
   Predicted **glass time** *after* “render”.

Process forward for Chrome

1. Make everything used passed in frame time. Then at;

   Last **glass time** *before* "render".

   This is where I spent a lot of my time

2. Make vsync source stable / add glass time predictor.
3. Pass in "glass time". Then at;

   Predicted **glass time** *after* "render".

Process forward for Chrome

1. Make everything used passed in frame time. Then at;

   Last **glass time** *before* “render”.

2. Make vsync source stable (add glass time predictor.

   I have code for this but haven’t landed

3. Pass in “glass time”. Then at;

   Predicted **glass time** *after* “render”.

Process forward for Chrome

1. Make everything used passed in frame time. Then at;

   Last **glass time** *before* "render".
2. Make vsync source stable / add glass time predictor.
3. Pass in "glass time". Then

   Predicted **glass time**

What to do next

http://goo.gl/MHGWTc

How can you help?

- Don’t use Now()!
- Don’t assume render time is in the past.

http://goo.gl/MHGWTc

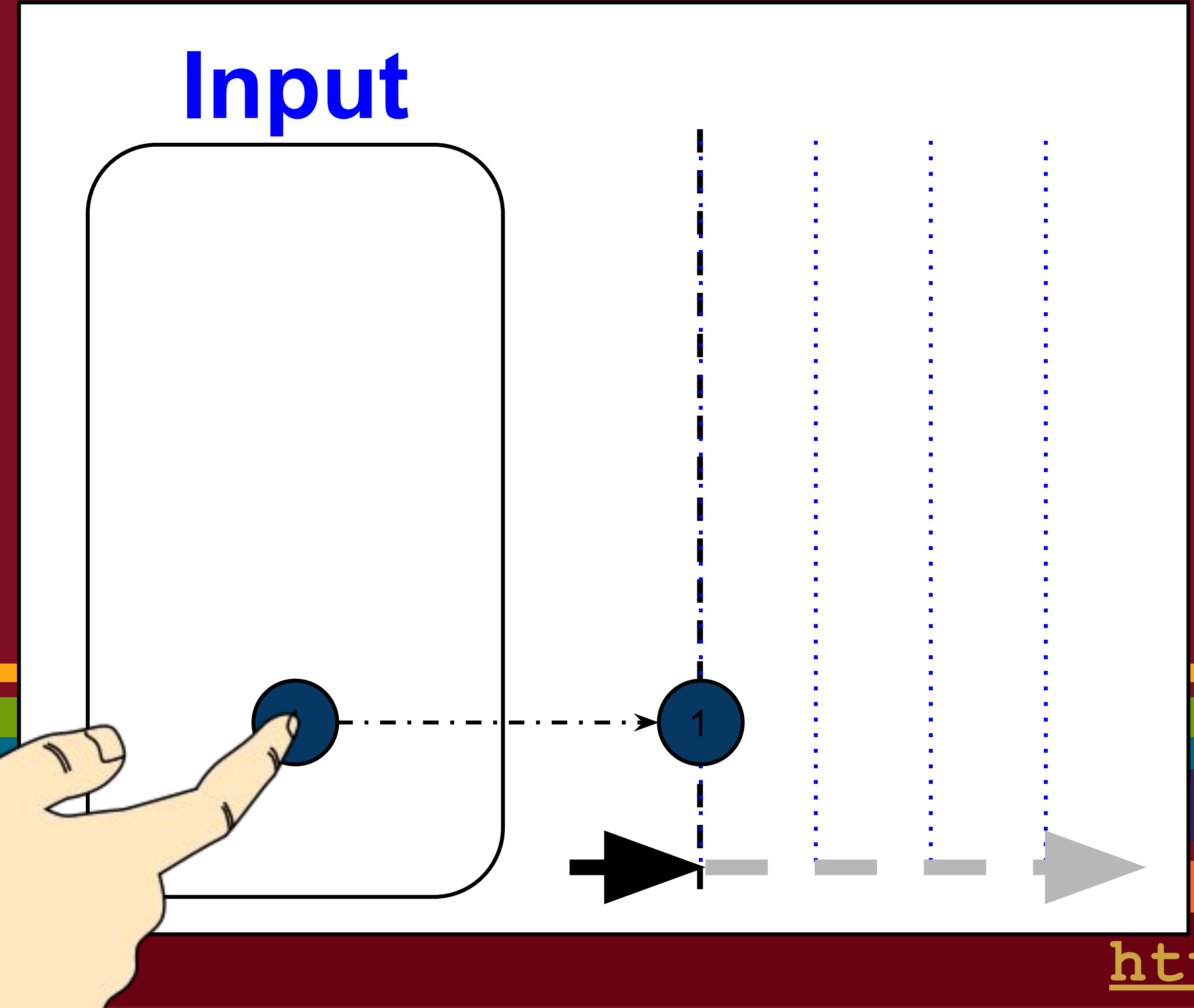
Input and “glass time”

Predicting how a user interacts

http://goo.gl/MHGWTc

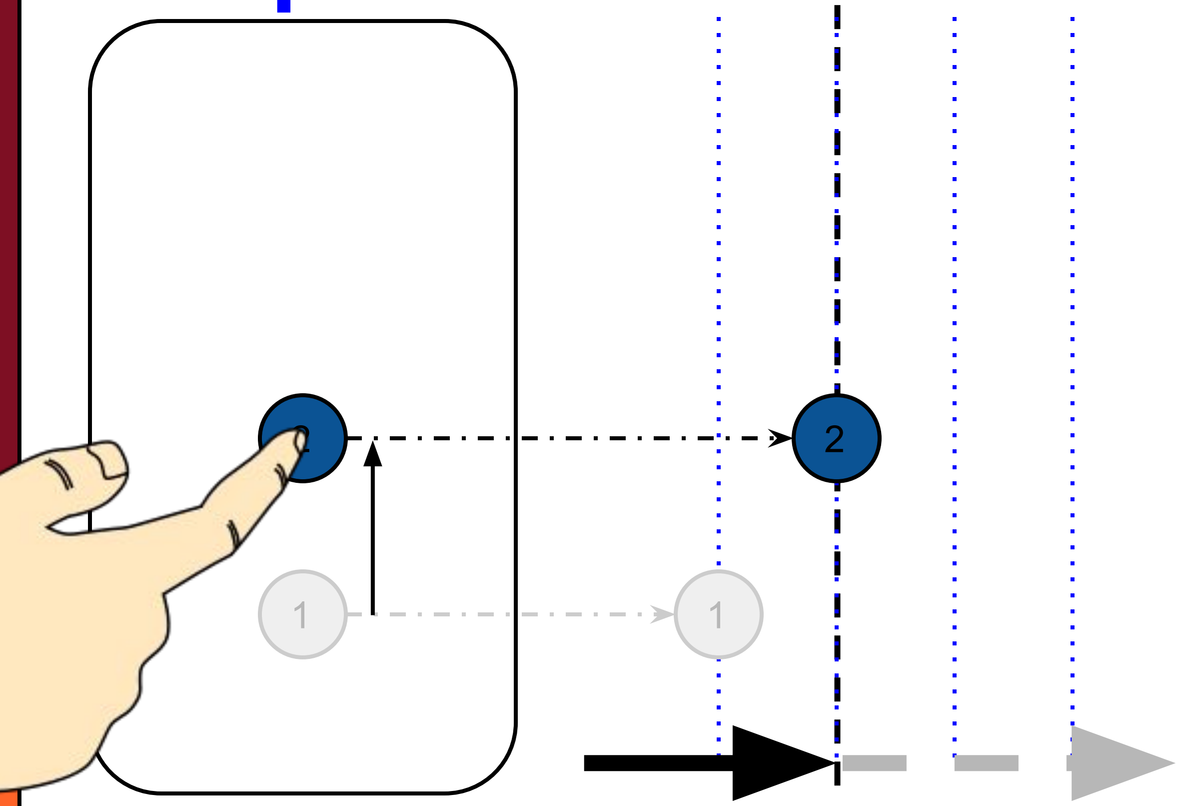
Experiment 2 - Finger Tracking

Try and draw a ball under a person's finger.



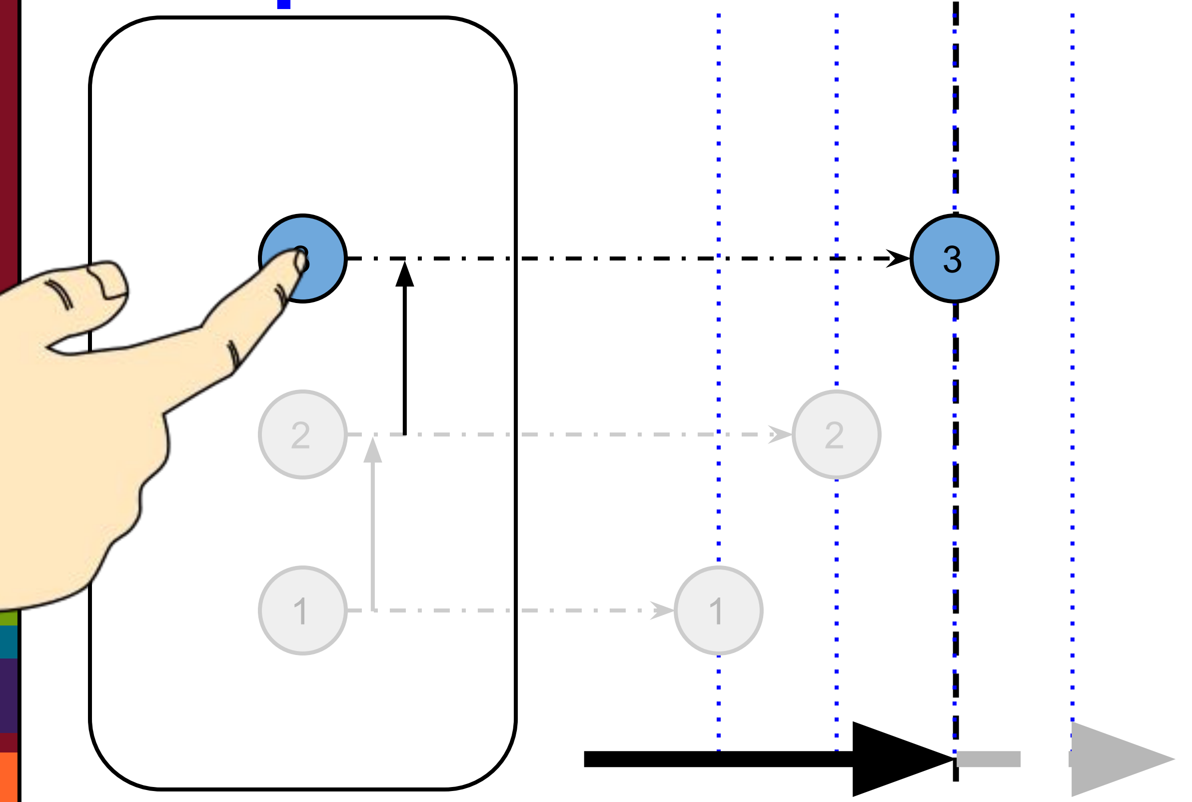
http://goo.gl/MHGWTc

Input



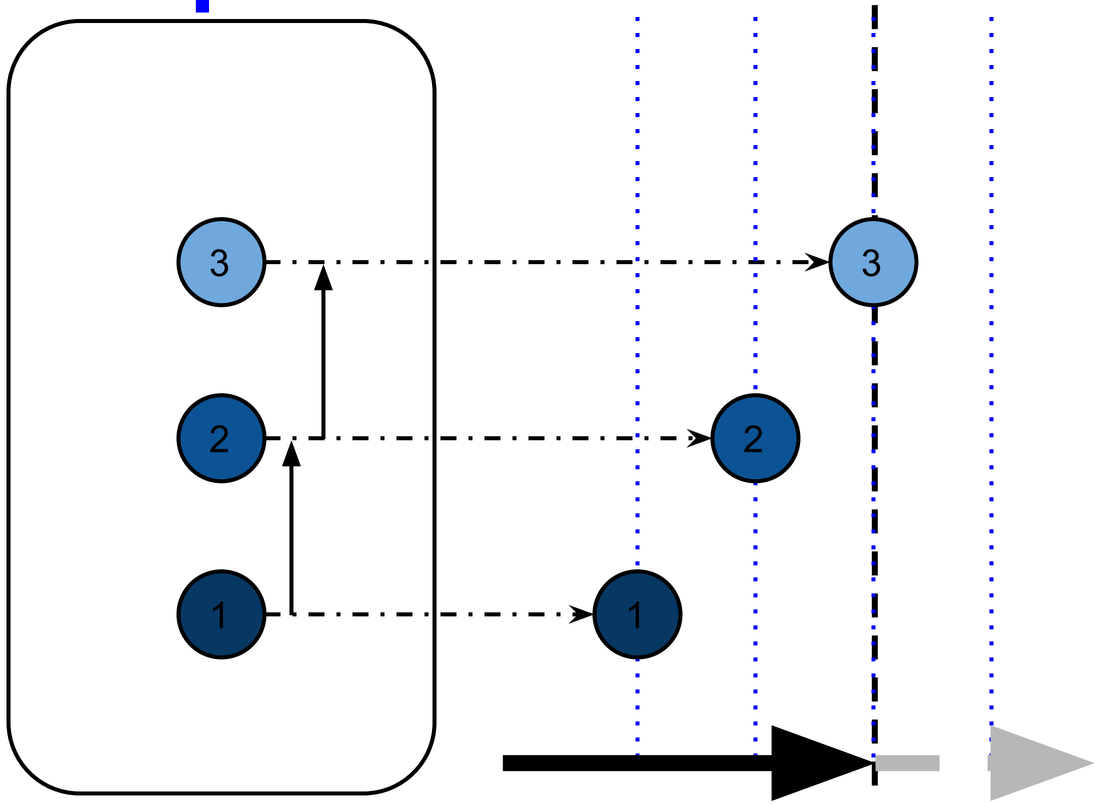
http://goo.gl/MHGWTc

Input



http://goo.gl/MHGWTc

Input

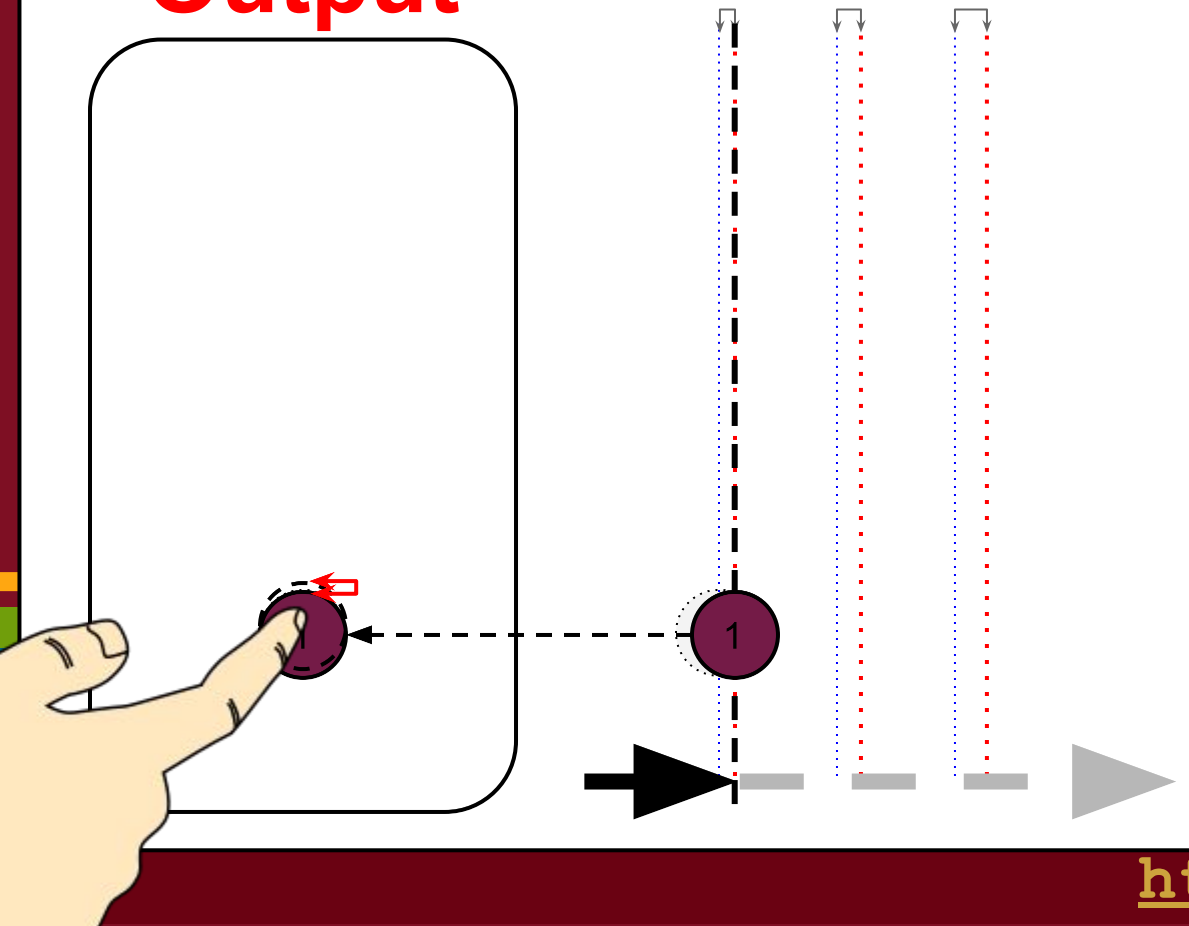


http://goo.gl/MHGWTc

Effect of lag on output

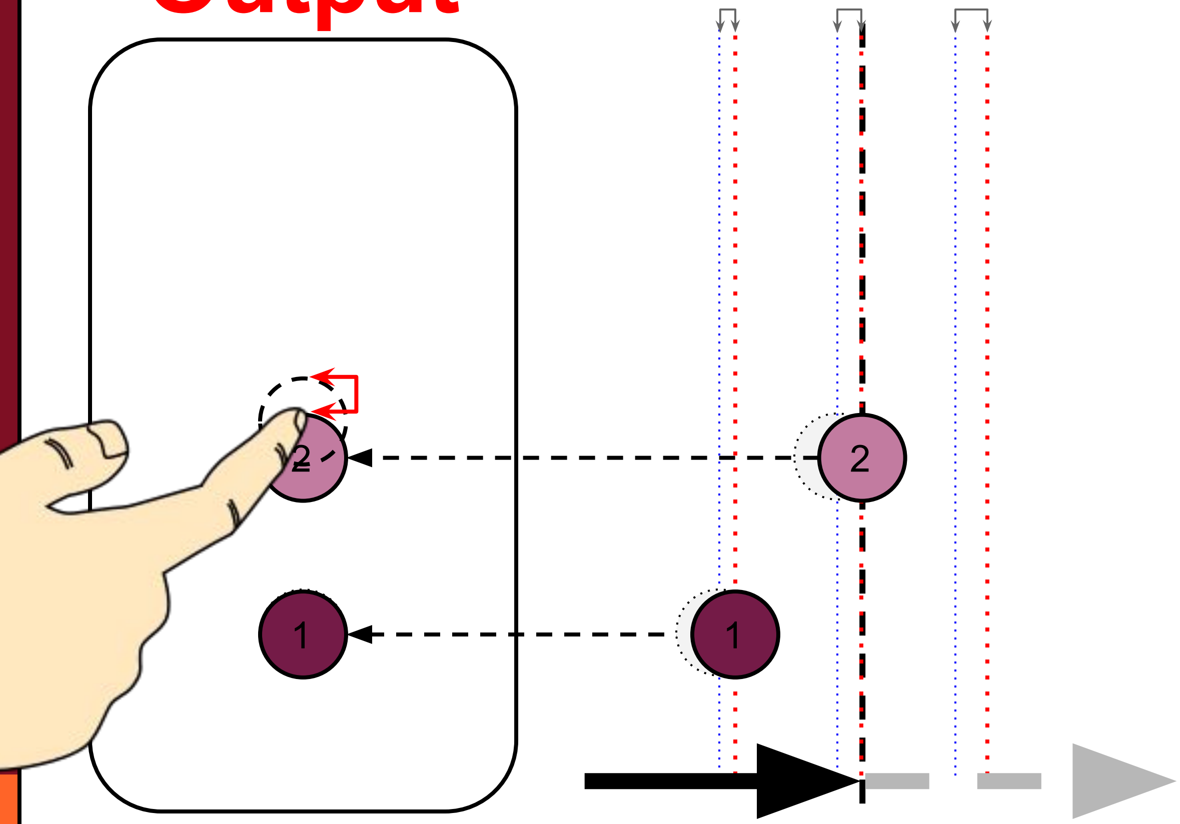
http://goo.gl/MHGWTc

Output



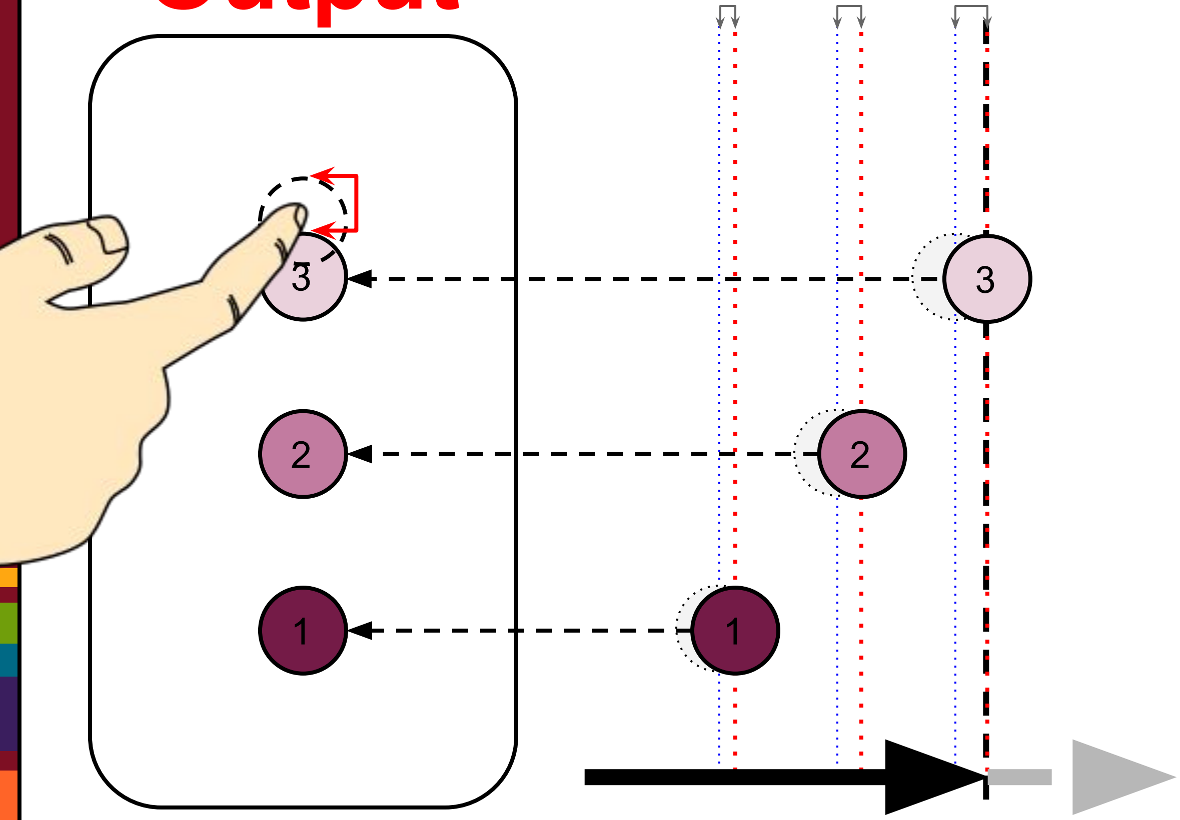
http://goo.gl/MHGWTc

Output



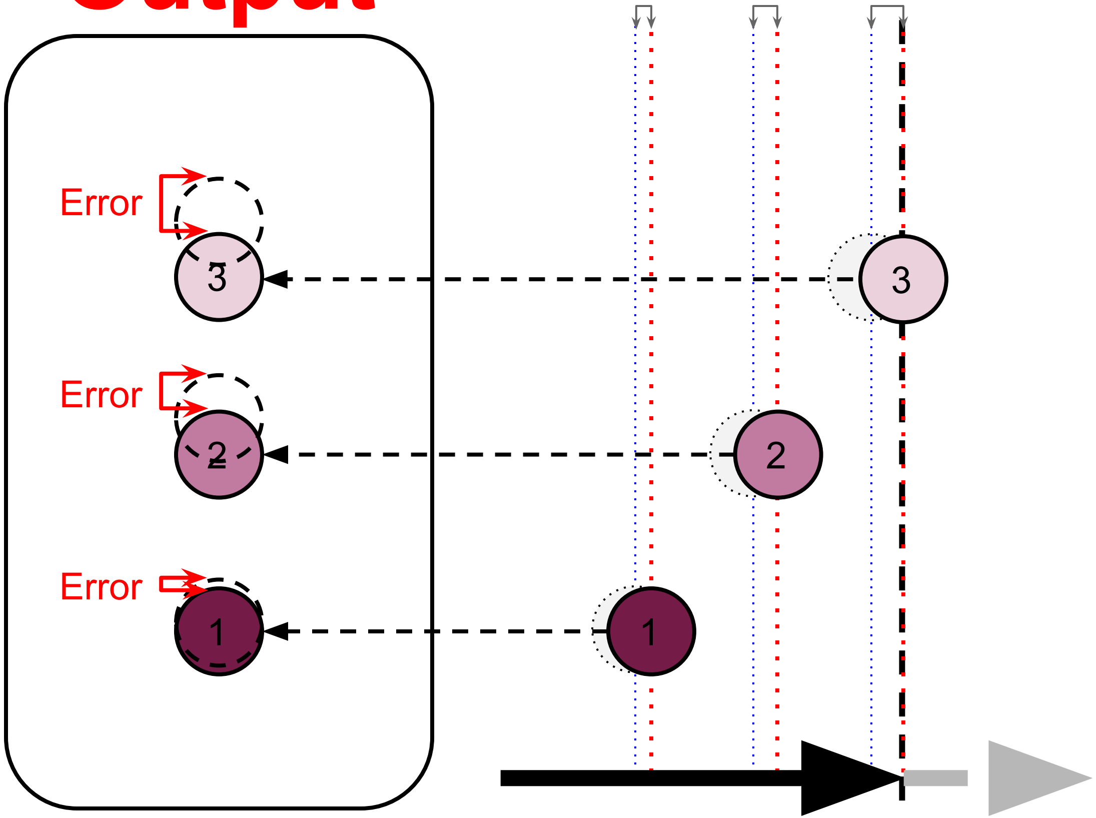
http://goo.gl/MHGWTc

Output



http://goo.gl/MHGWTc

Output



http://goo.gl/MHGWTc

Error caused by

The input/output error is caused by;

- render **lag**
- display **lag**
- input **lag**
- vsync and input frequency differences

Error caused by

The input/output error is caused by;

- render **lag**
- display **lag**
- input **lag**
- vsync and input frequency differences

Seen previously

Seen previously

Error caused by

The input/output error is caused by;

- render **lag**
- display **lag**
- input **lag**
- vsync and inpu[illegible]y differences

New!

New!

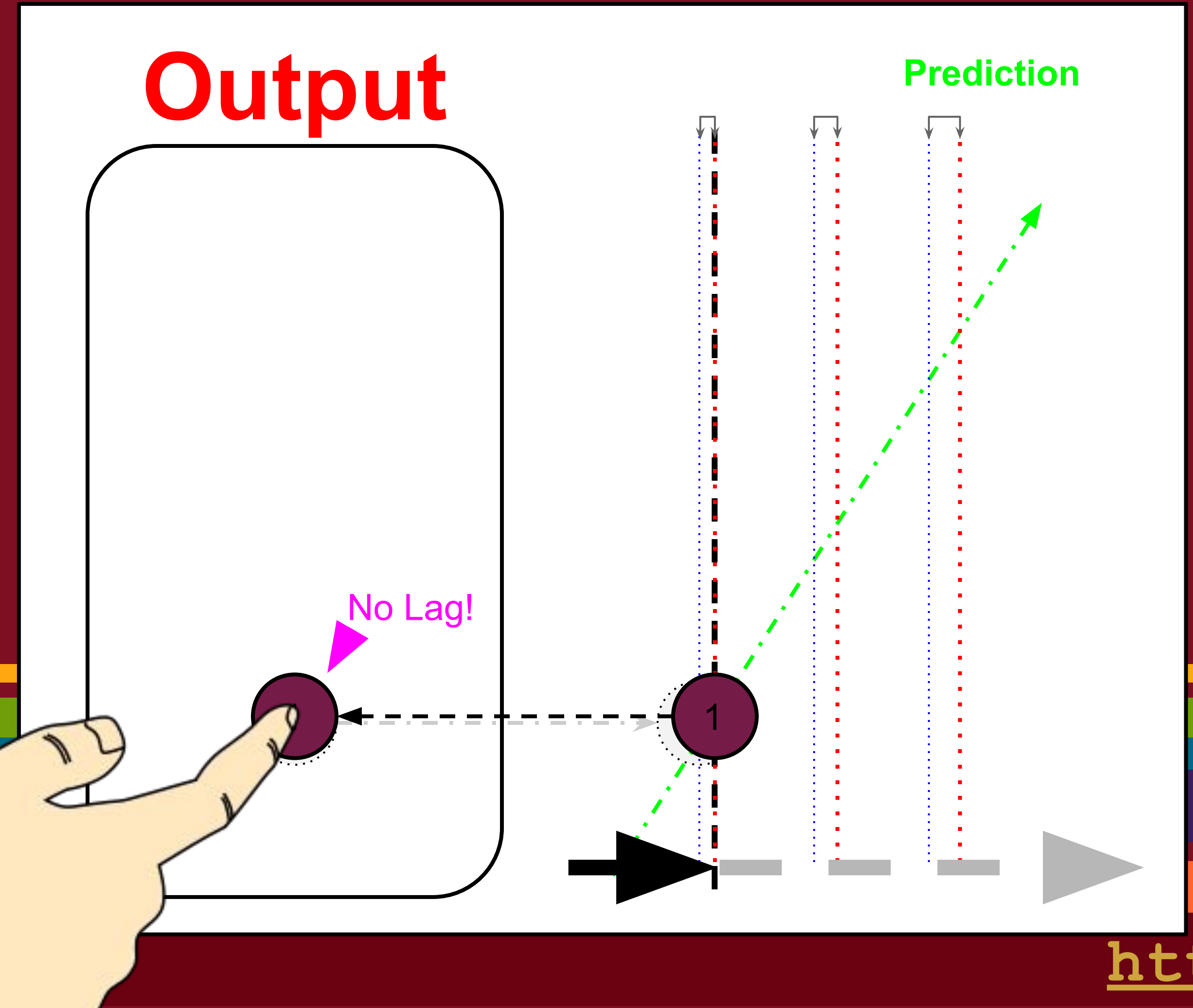
Sum of **Lag** == Latency

- Like previously we can solve lag by rendering for time in future.

- Complicated by fact that input is not deterministic. **But** is predictable over short periods.

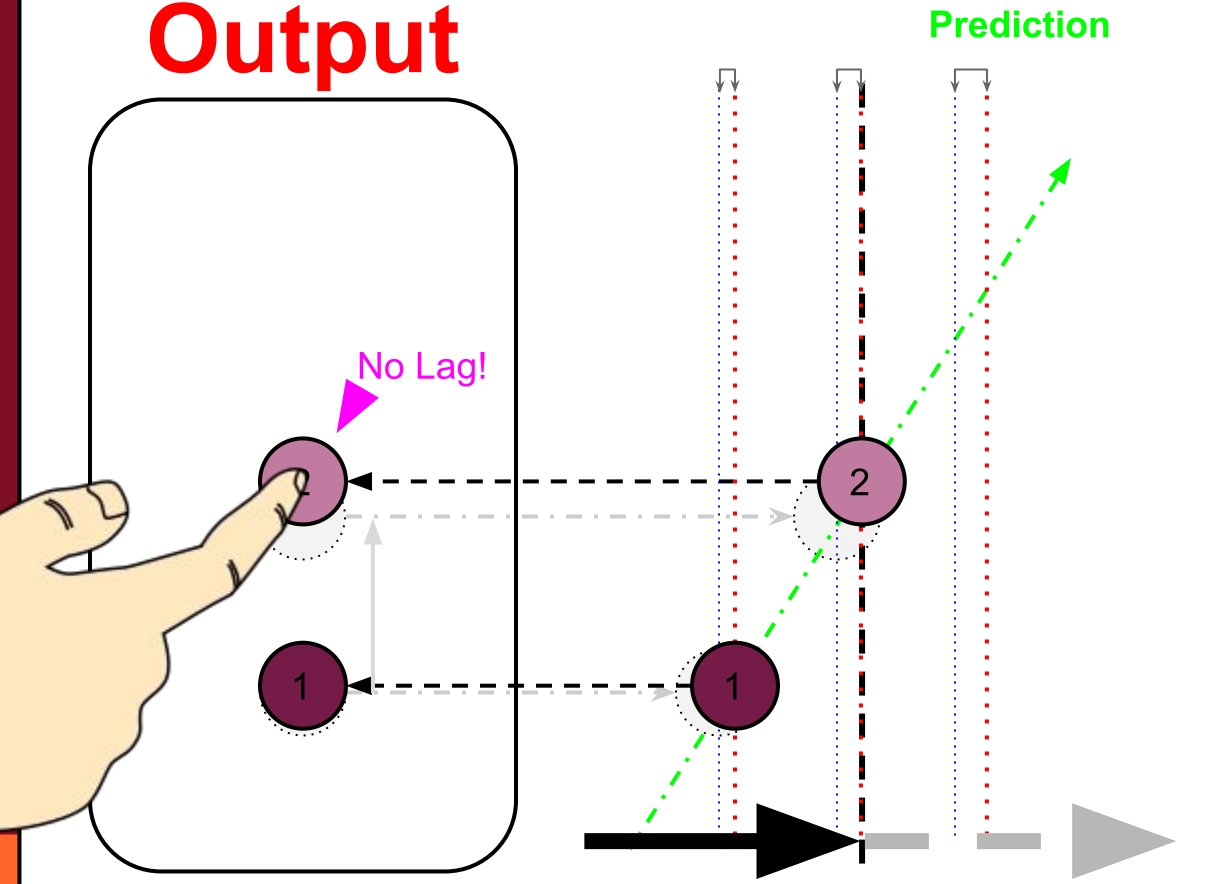
Output with Prediction

http://goo.gl/MHGWTc



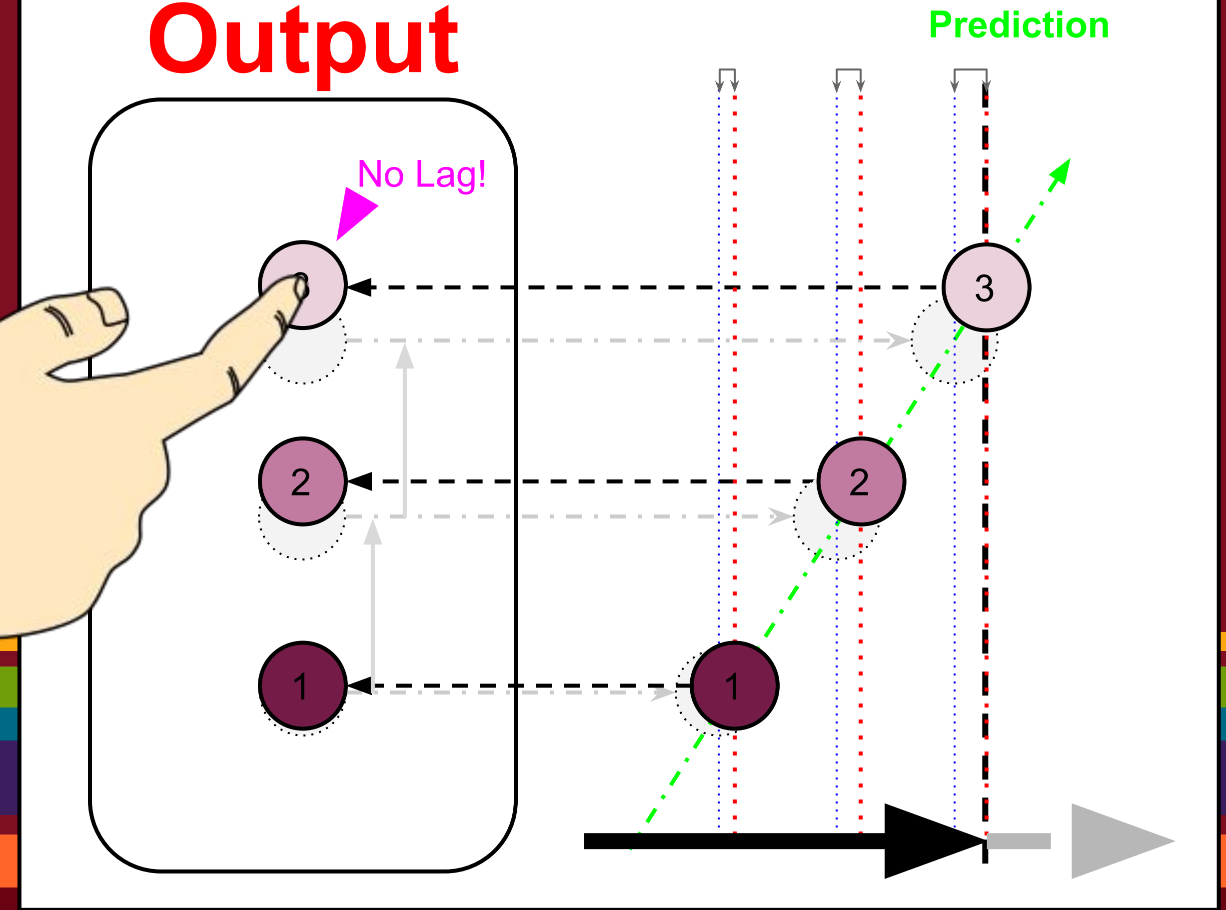
http://goo.gl/MHGWTc

Output



http://goo.gl/MHGWTc

Output



http://goo.gl/MHGWTc

Prediction

- The real world obeys physics.
- Humans like the physics.
- ??????

More input, better outcome

The more data, the better it works!

Error caused by

The input/output error is caused by;

- render **lag**
- display **lag**
- input **lag**
- vsync and inpu … erences

Can be reduced

Can **not** be removed

Can be reduced

Can be removed

Error caused by

The input/output error is caused by;

- render **lag**
- display **lag**
- input **lag**
- vsync and input frequency differences

Can **not** be removed

Can't eliminate all sources of **lag**

Hence,

You always prediction!

With prediction

Error is **error in your prediction system**.

For steady state, **this can be zero**.

You can improve this!

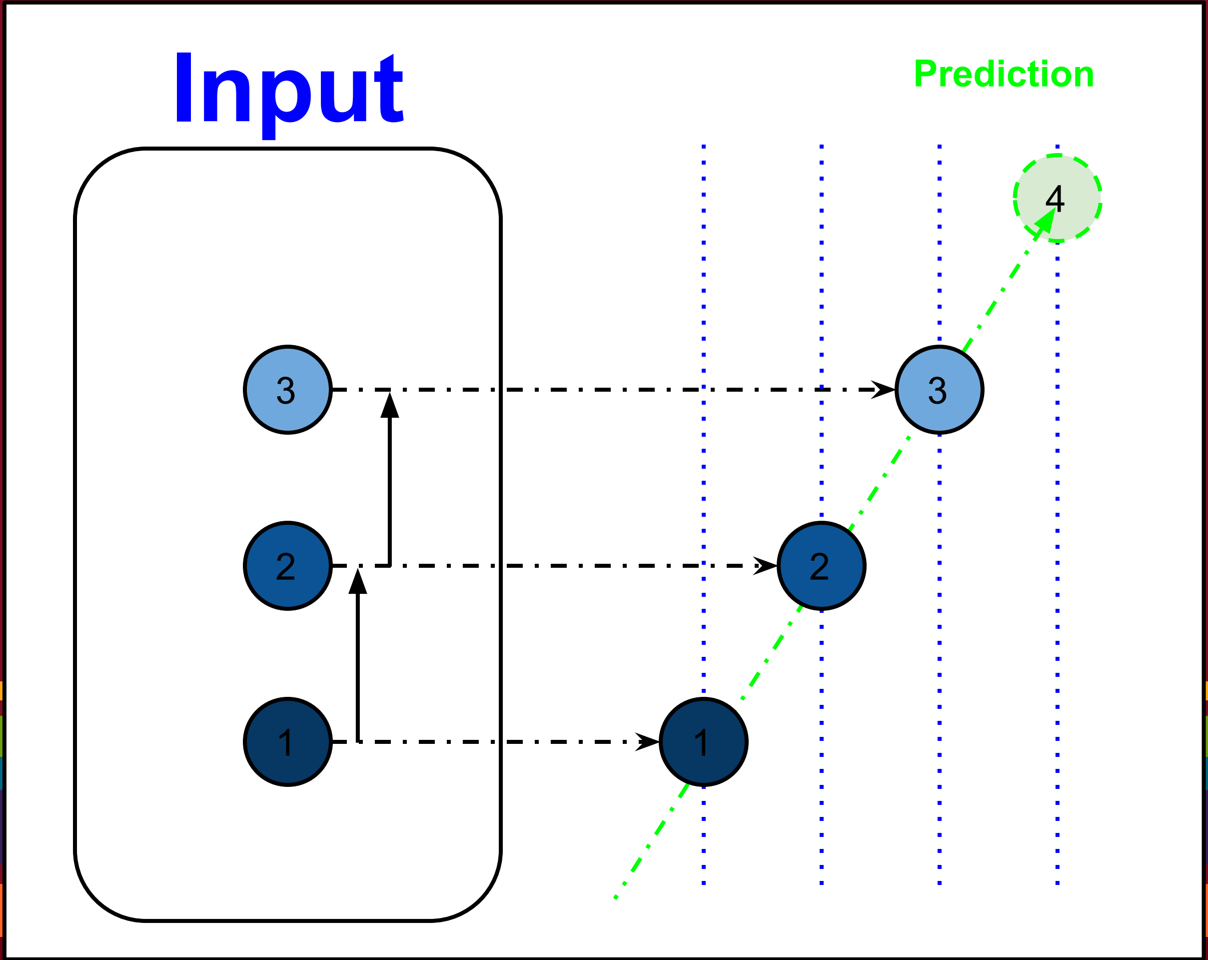
Other advantages of input prediction

http://goo.gl/MHGWTc

Decoupled input and output

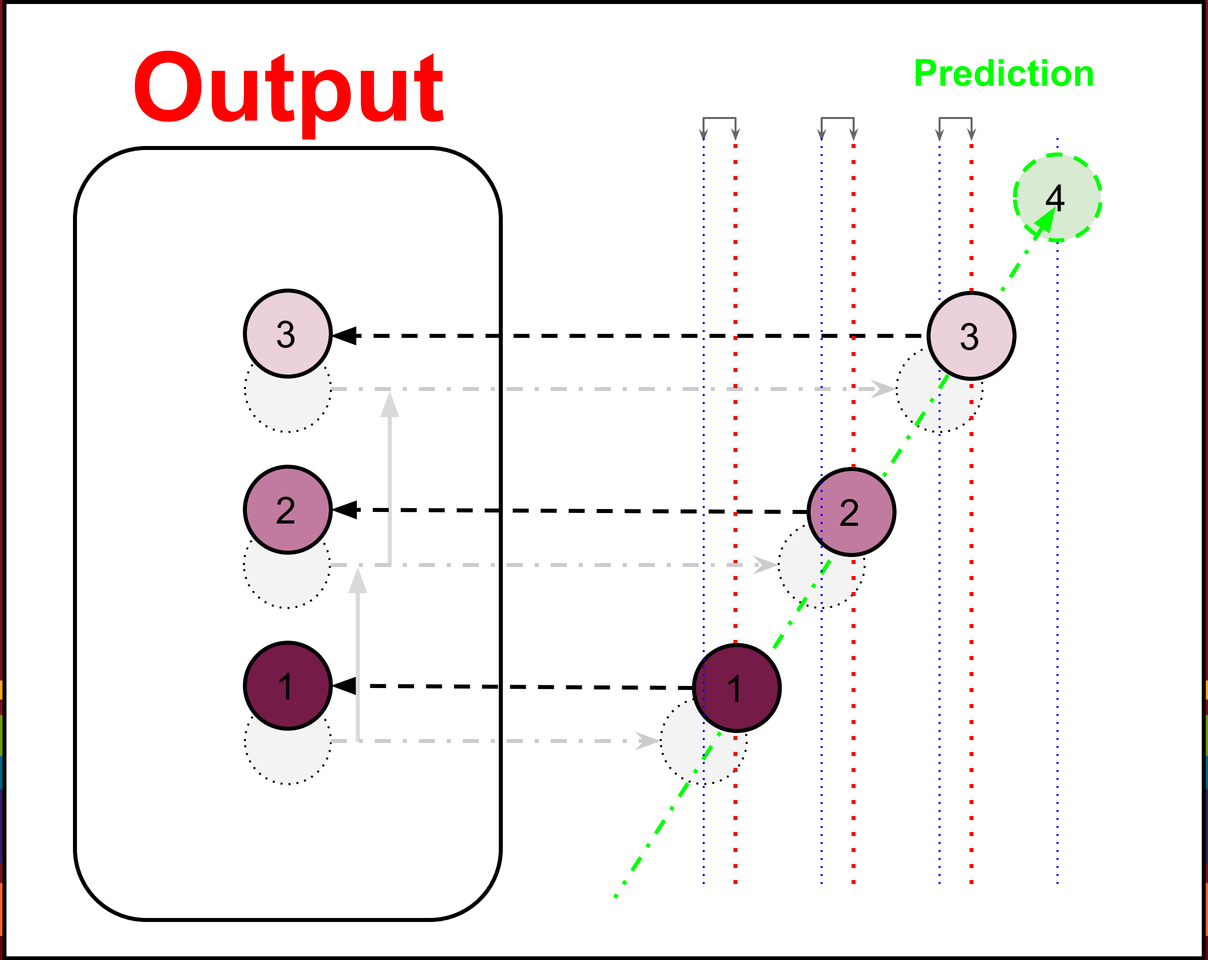
Once you have a prediction system your input and output systems are totally decoupled. This gives you some nice properties;

- Deal with nonlinear input
- Deal with missing input
- Deal with noisy input



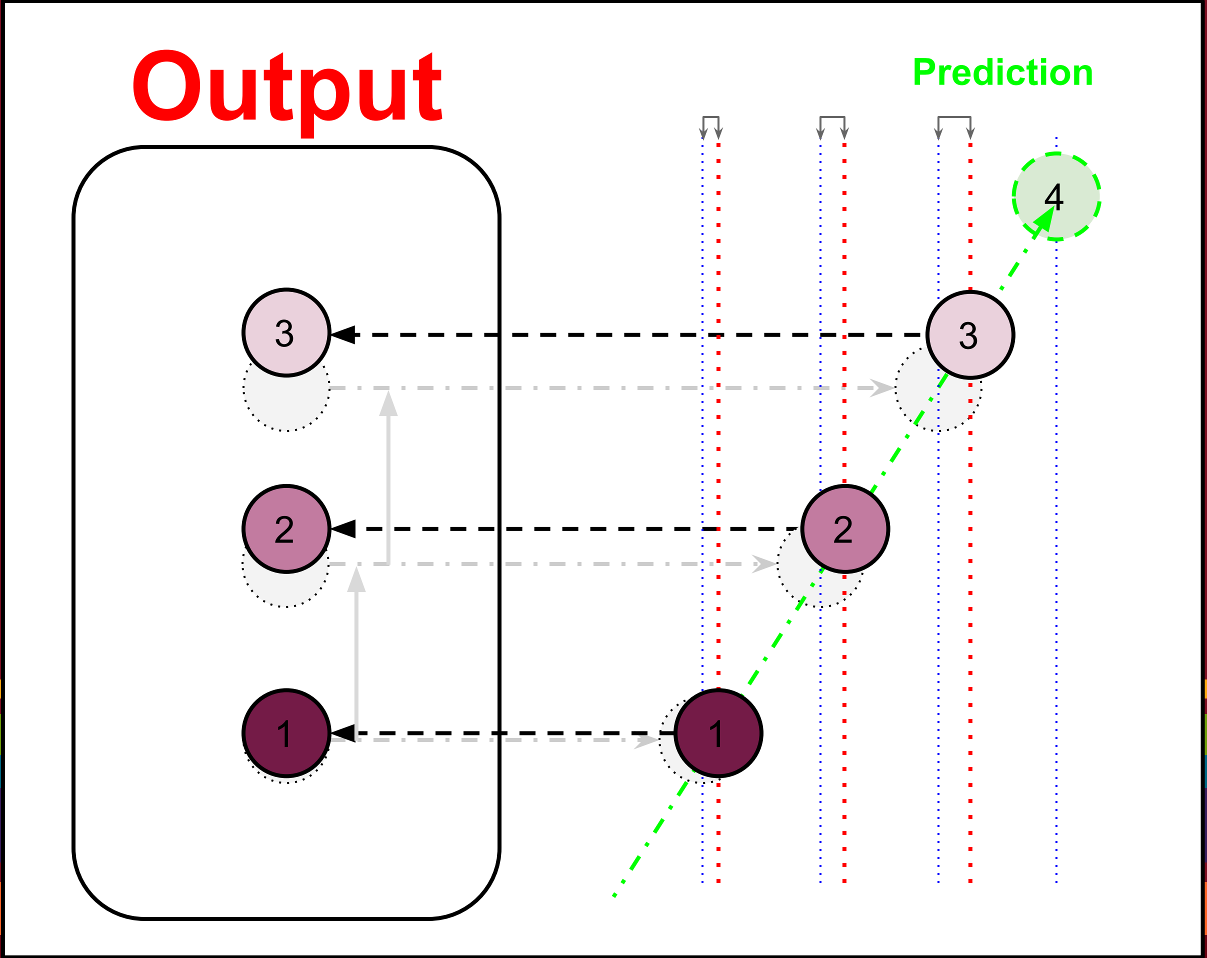
http://goo.gl/MHGWTc

Constant delay

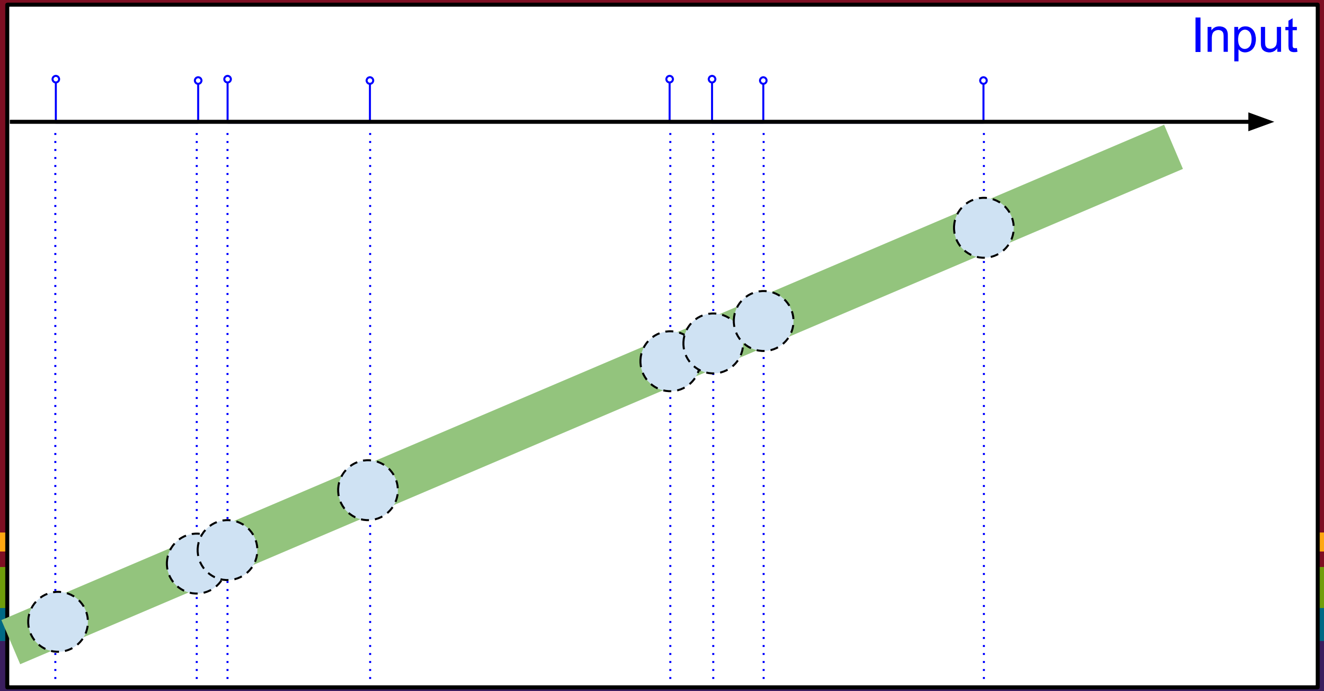


http://goo.gl/MHGWTc

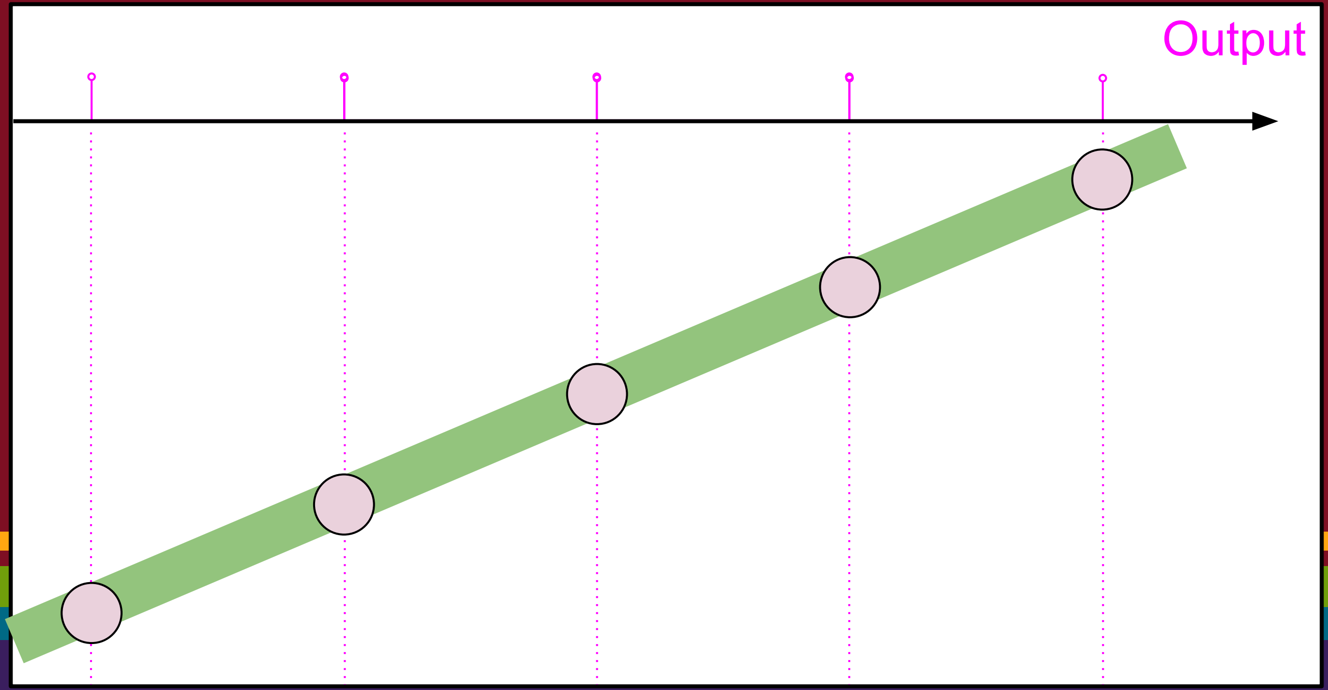
Variable delay



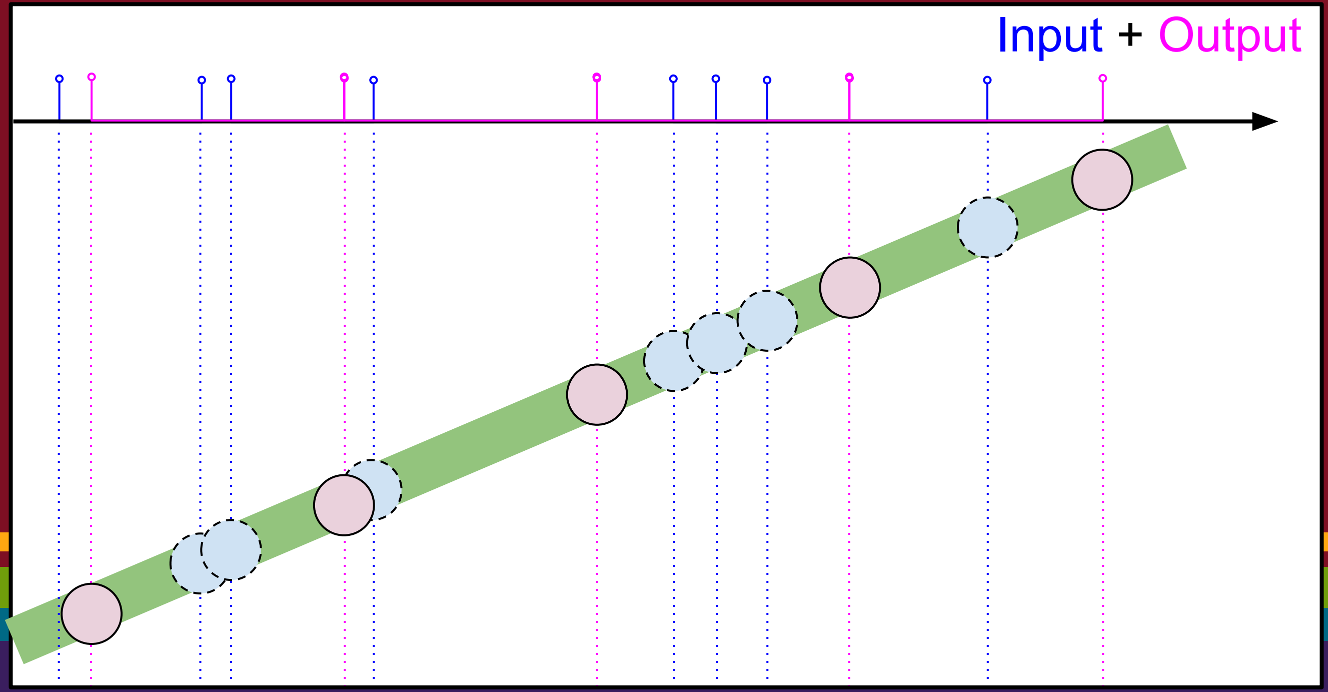
http://goo.gl/MHGWTc



http://goo.gl/MHGWTc



http://goo.gl/MHGWTc



http://goo.gl/MHGWTc

Stop here

Color meani

Red Background for generic

Experiment slides?

- Black is for time indicators
- Red is for errors
- Purple is for lag

Blues for physical world

Mauve for rendered output

Timeline slides?

- Black is for tim
- Red is for time
- Blue is for time
- Purple is for lag
- Yellow is for tas

http://goo.gl/MHGWTc

Constant Lag

Why longer lag which is consistent is better...

Animations and people

To understand how **lag** effects **animations**, you first need to understand how the human visual perception system works.

Humans are great predictors

It was vital for our survival to be able to;

- **predicting movement**
  of physical objects
- **detect subtle differences**
  in the movement

Humans are great predictors

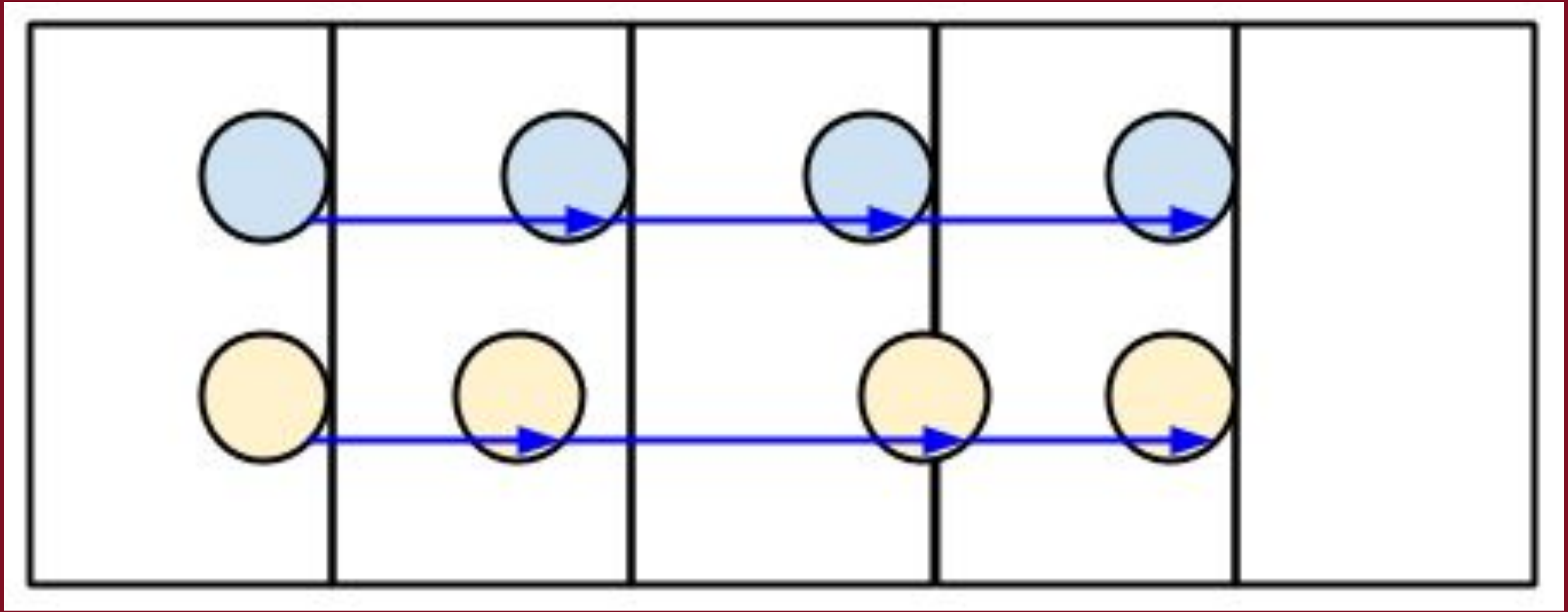
People can detect small and subtle issues in the position of moving objects

People find these issues **disturbing and uncomfortable**.

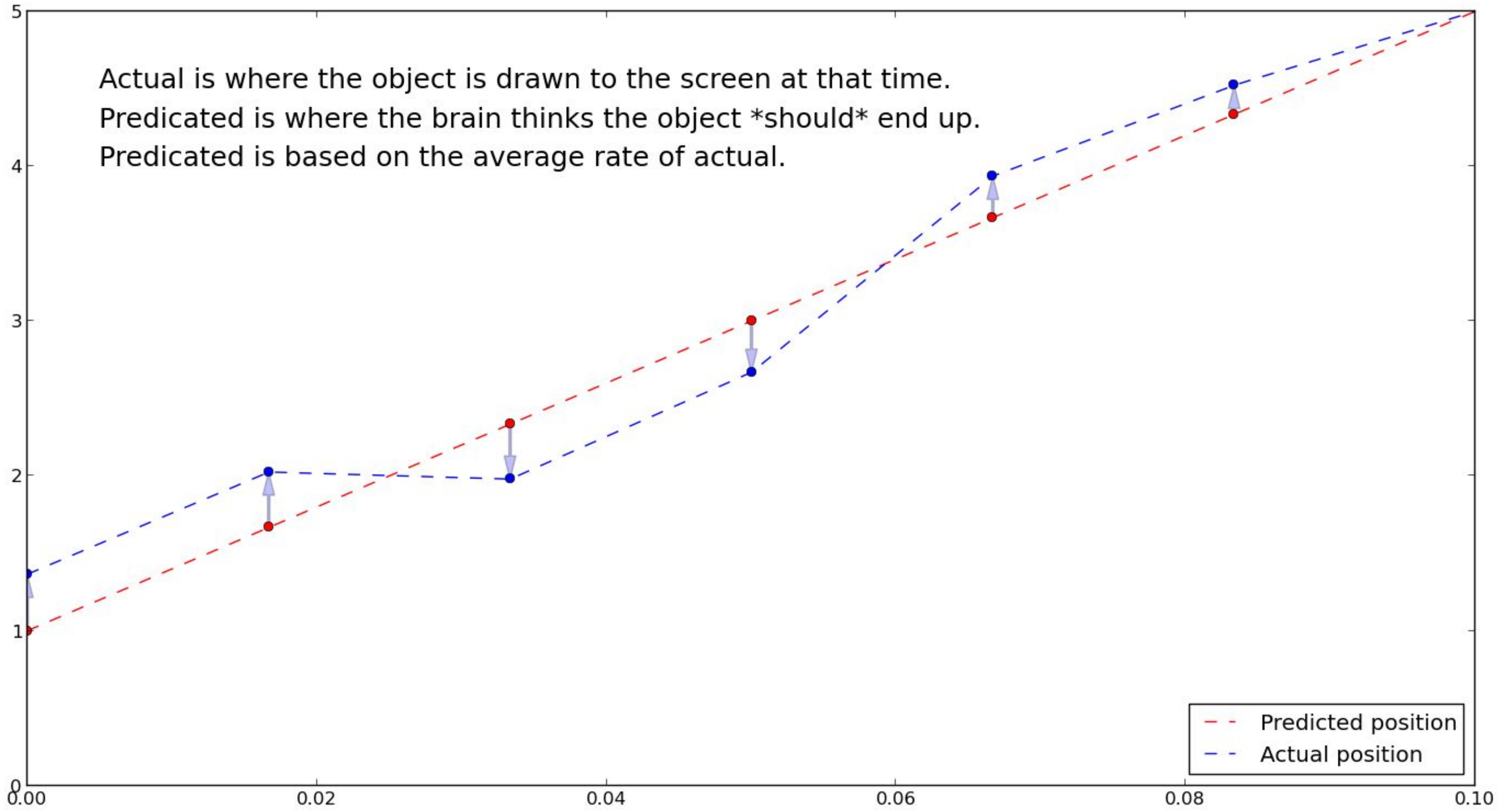
jitter == abs(Prediction - Actual)

Difference between,

predicted position
and
actual position



http://goo.gl/MHGWTc



http://goo.gl/MhGWfc

How is time involved?

In animation systems,

position is **dependent** on time.

If we are rendering for the

wrong time,

Then the object will appear in the

wrong position.

Human Visual Perception

Lots of new research in 2010's, overturning 1980's "standards"

- Distinguish flashes/s (20ms // 48Hz)
- Recall specific images (13ms // 78Hz)
- See issues less than 0.6″ (arc seconds)

Compare to 60Hz // 16ms

BTW Human audio detects issues even below 2ms!

http://goo.gl/MHGWTc

Step 1 - Using “last display time”

Get the whole animation stack (compositor, renderer, etc) use the vsync signal

Then,

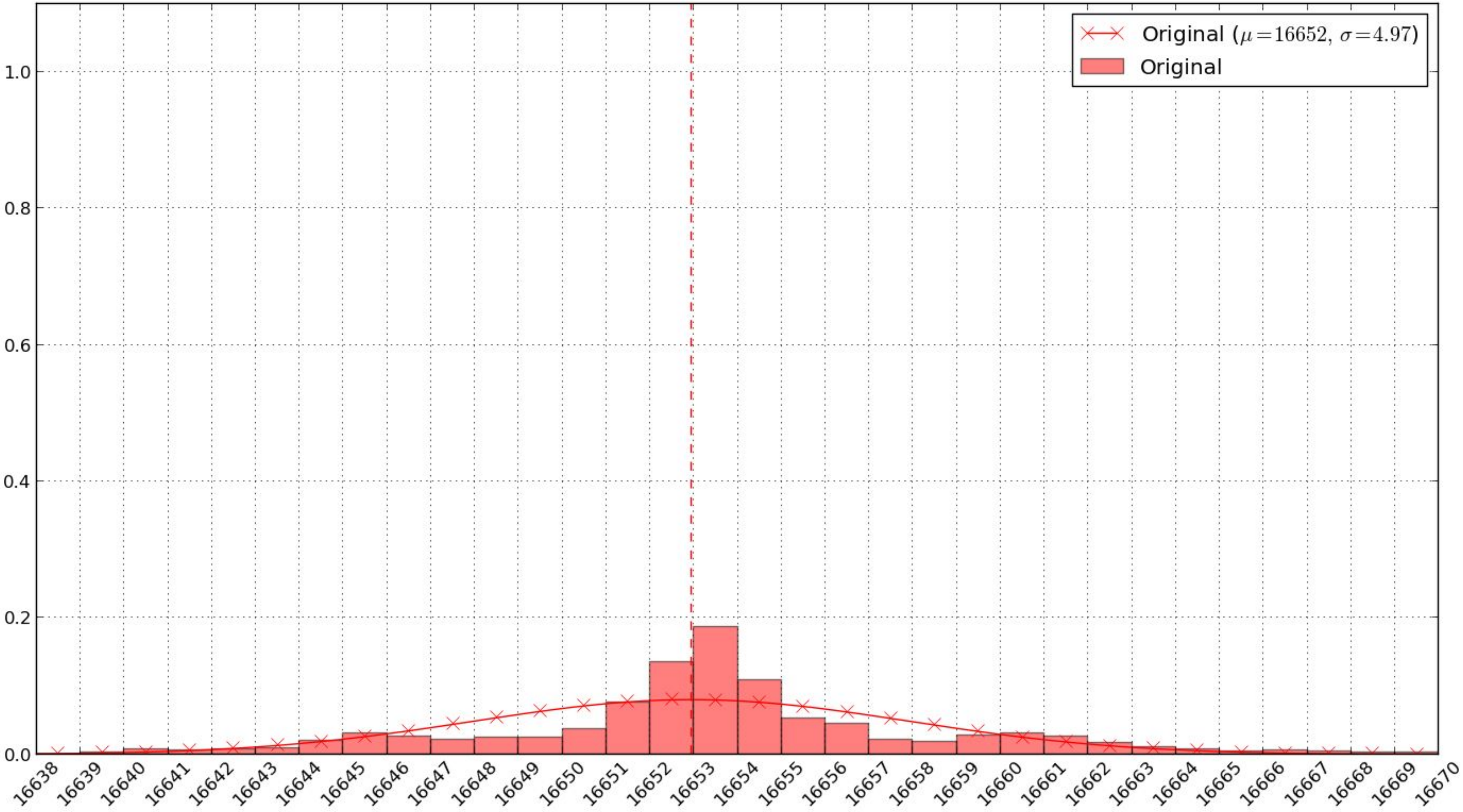
- RAF stddev == vsync stddev
- When no vsync, the stddev == 0.0000

Step 2a - Filter vsync

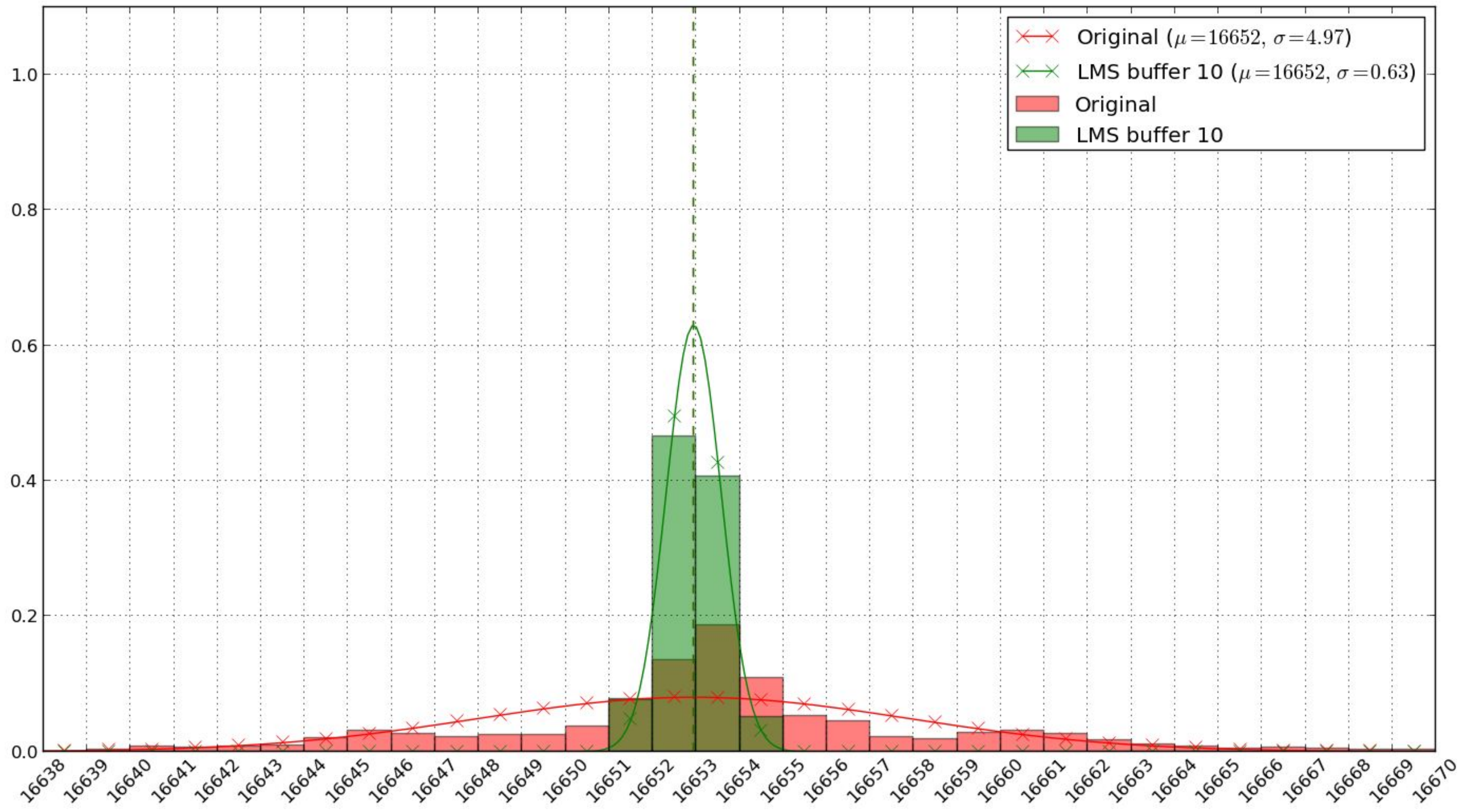
Get the whole animation stack (compositor, renderer, etc) use the vsync signal

Then,

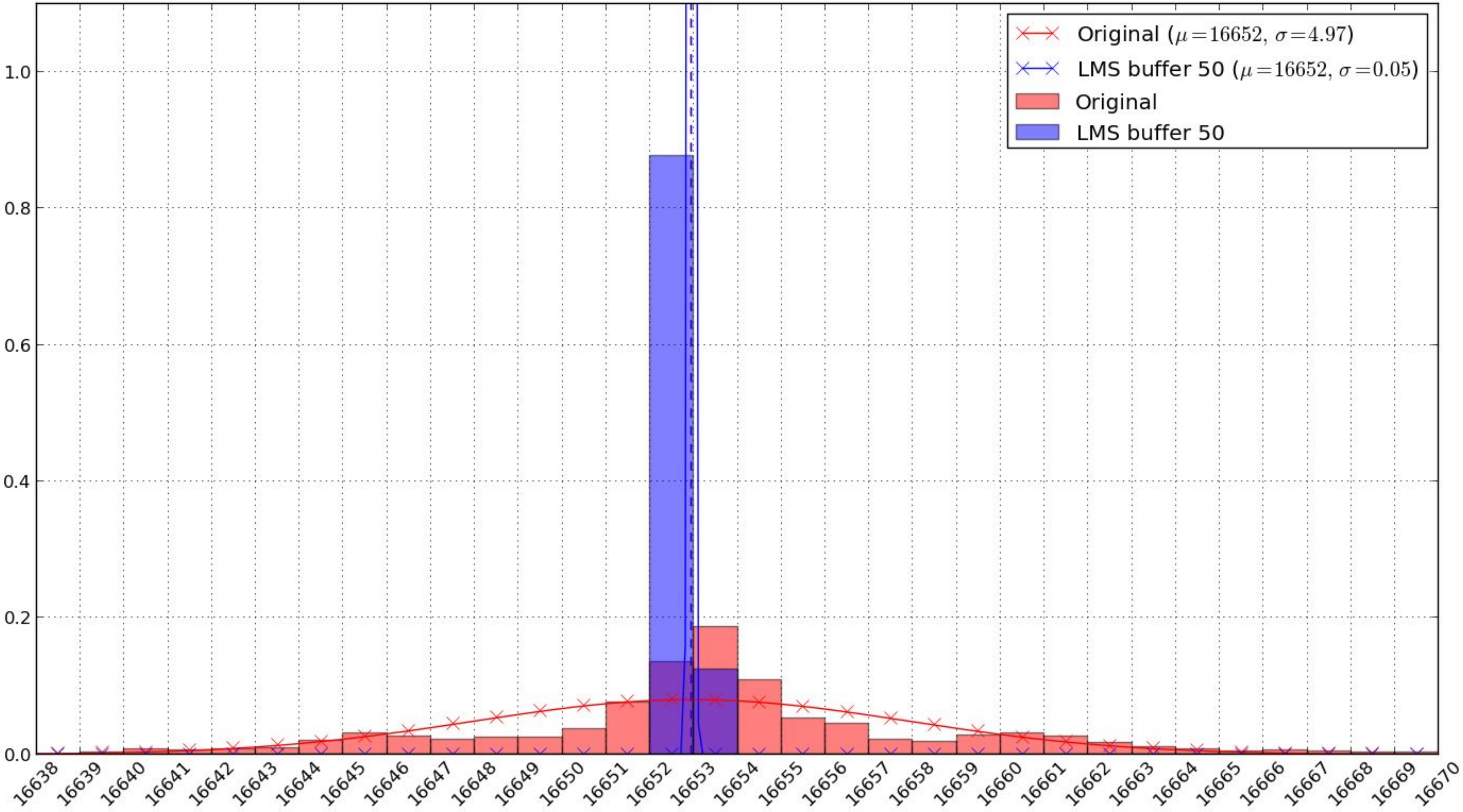
- RAF stddev == vsync stddev
- When no vsync, the stddev == 0.0000



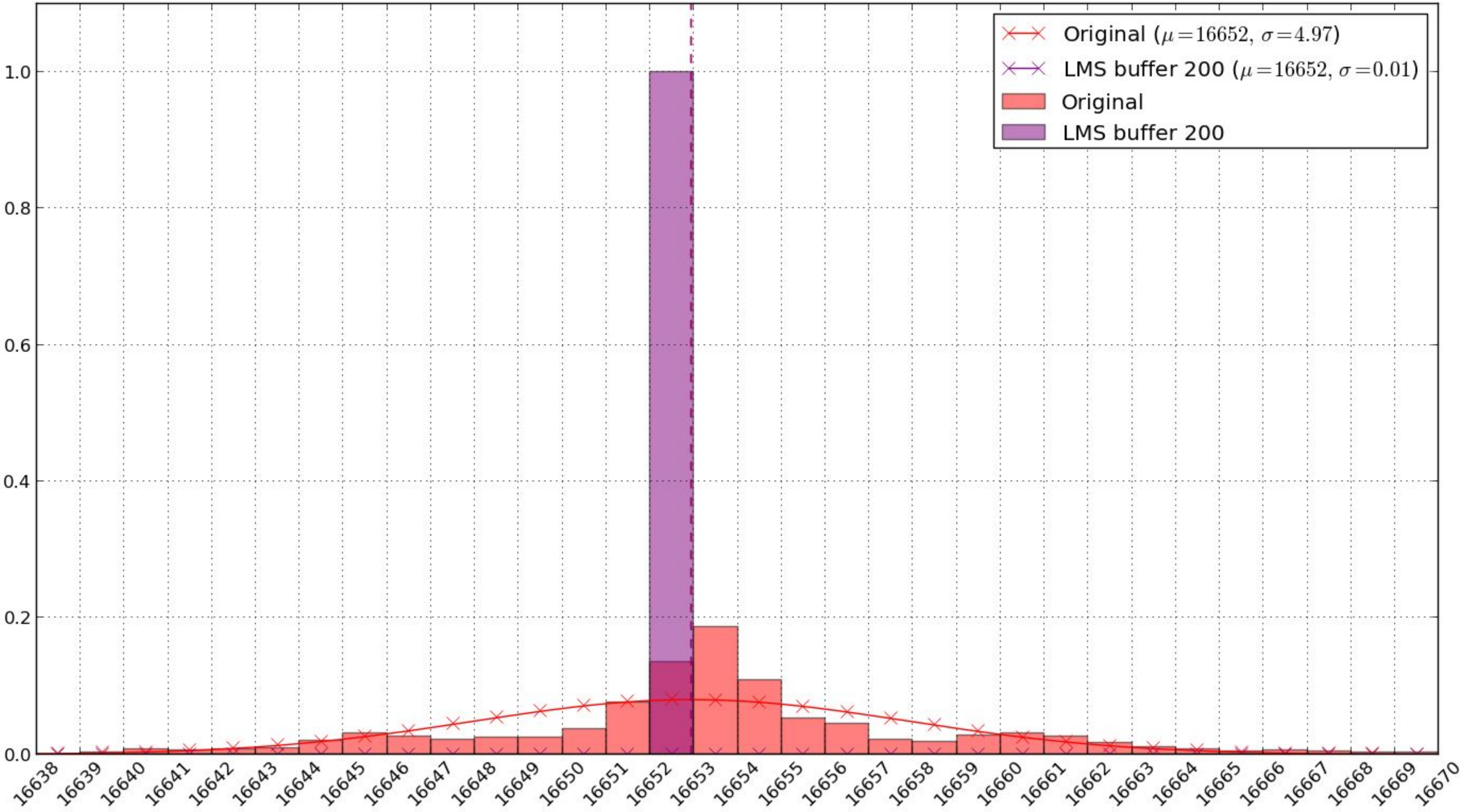
http://goo.gl/MnGWfc



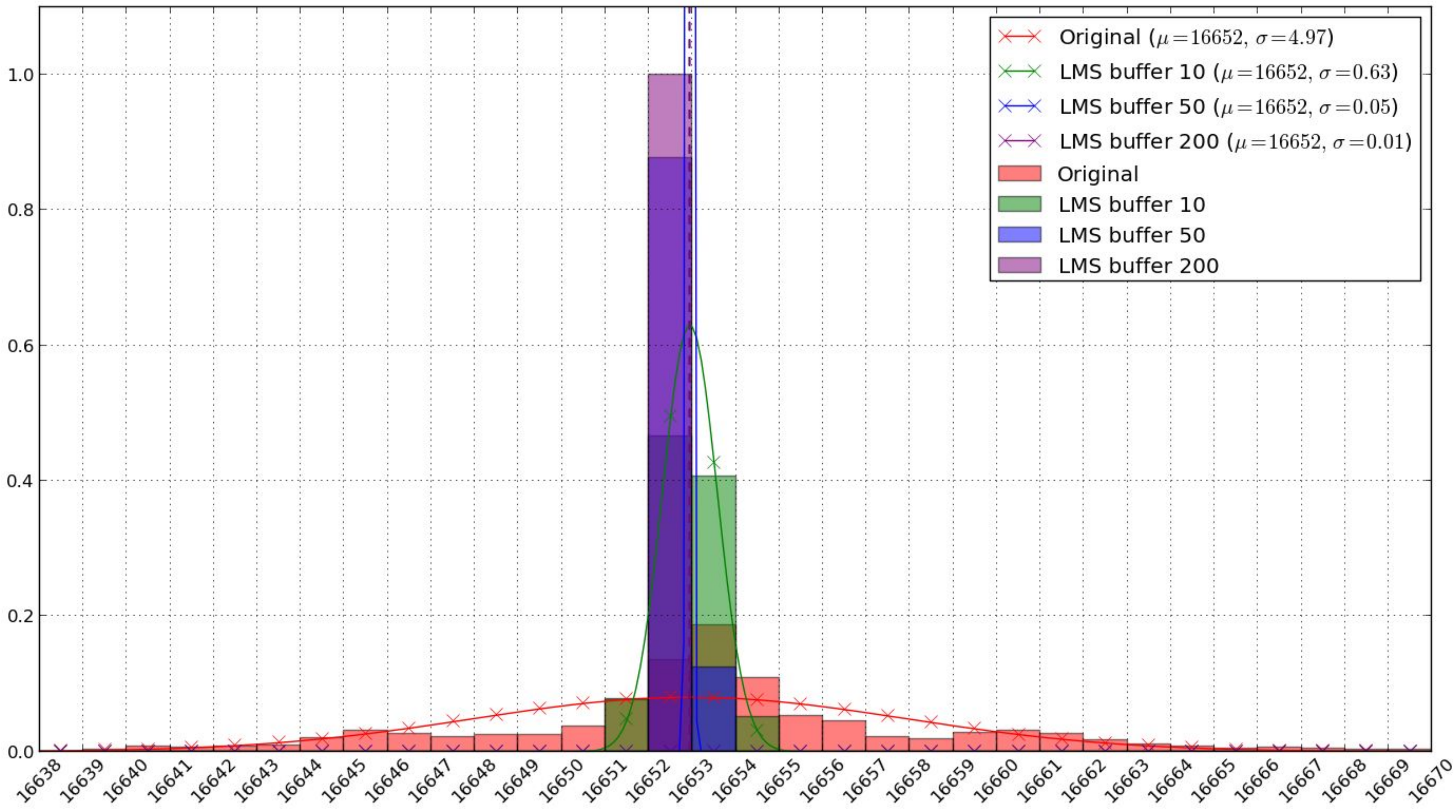
http://goo.gl/MnGWfc



http://goo.gl/MHGWfc



Original ($\mu = 16652$, $\sigma = 4.97$)
LMS buffer 200 ($\mu = 16652$, $\sigma = 0.01$)
Original
LMS buffer 200
0.0
0.2
0.4
0.6
0.8
1.0
16638
16639
16640
16641
16642
16643
16644
16645
16646
16647
16648
16649
16650
16651
16652
16653
16654
16655
16656
16657
16658
16659
16660
16661
16662
16663
16664
16665
16666
16667
16668
16669
16670



Original ($\mu = 16652$, $\sigma = 4.97$)
LMS buffer 10 ($\mu = 16652$, $\sigma = 0.63$)
LMS buffer 50 ($\mu = 16652$, $\sigma = 0.05$)
LMS buffer 200 ($\mu = 16652$, $\sigma = 0.01$)
Original
LMS buffer 10
LMS buffer 50
LMS buffer 200
0.0
0.2
0.4
0.6
0.8
1.0
16638
16639
16640
16641
16642
16643
16644
16645
16646
16647
16648
16649
16650
16651
16652
16653
16654
16655
16656
16657
16658
16659
16660
16661
16662
16663
16664
16665
16666
16667
16668
16669
16670

http://goo.gl/MHGWTc

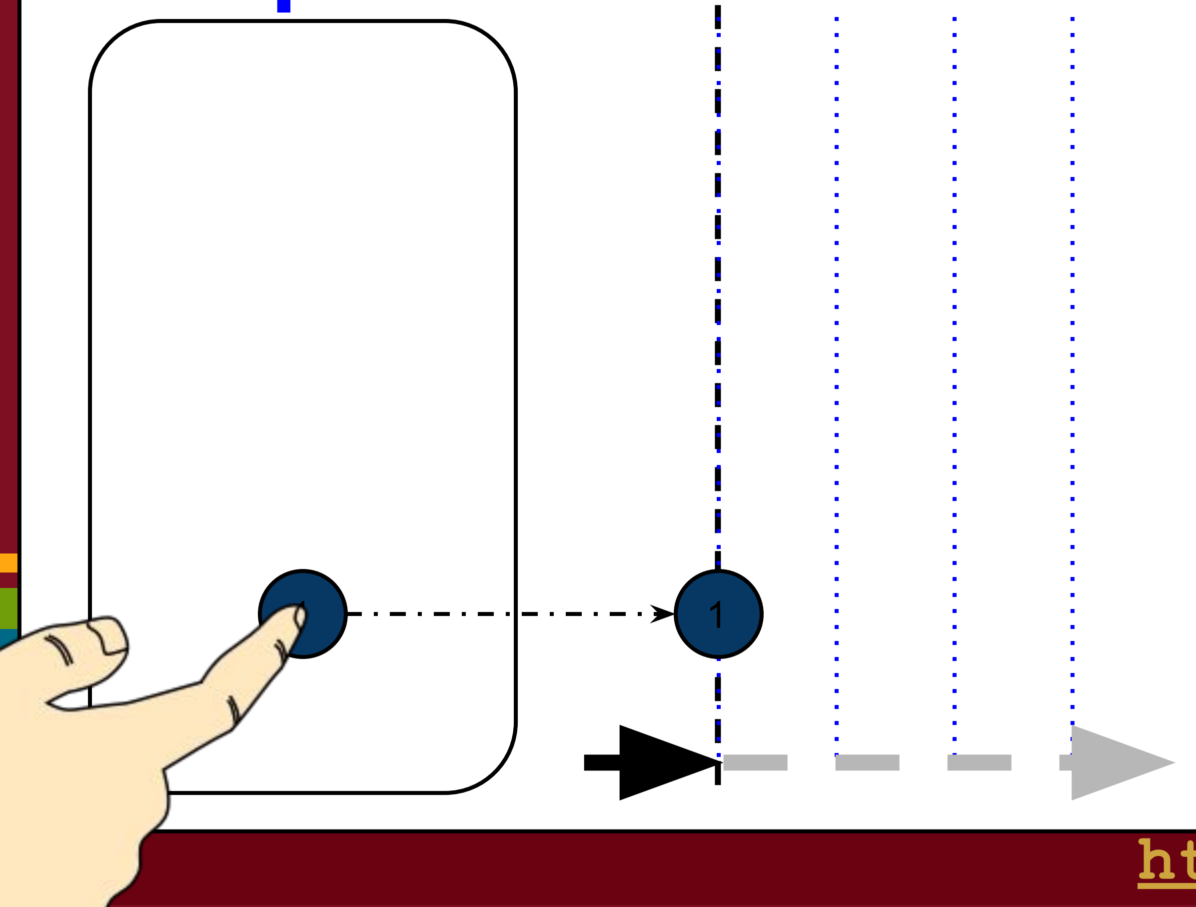
How is time involved?

Movement is

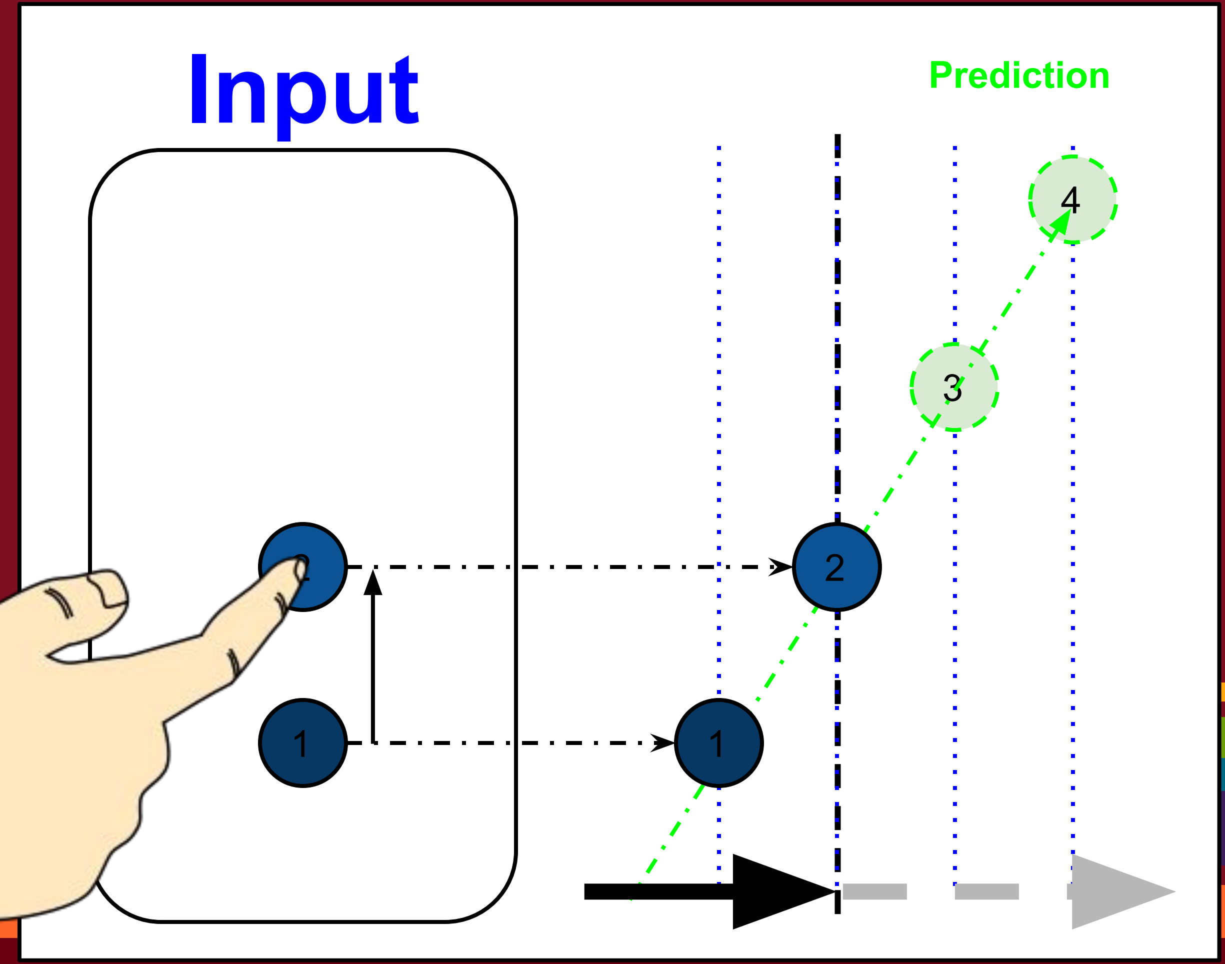
$$x(t_n) = x(t_{-1}) + \Delta x * (t_n - t_{n-1})$$

If we are rendering for the wrong time, no matter how fast we render, the animation will **still appear janky**.

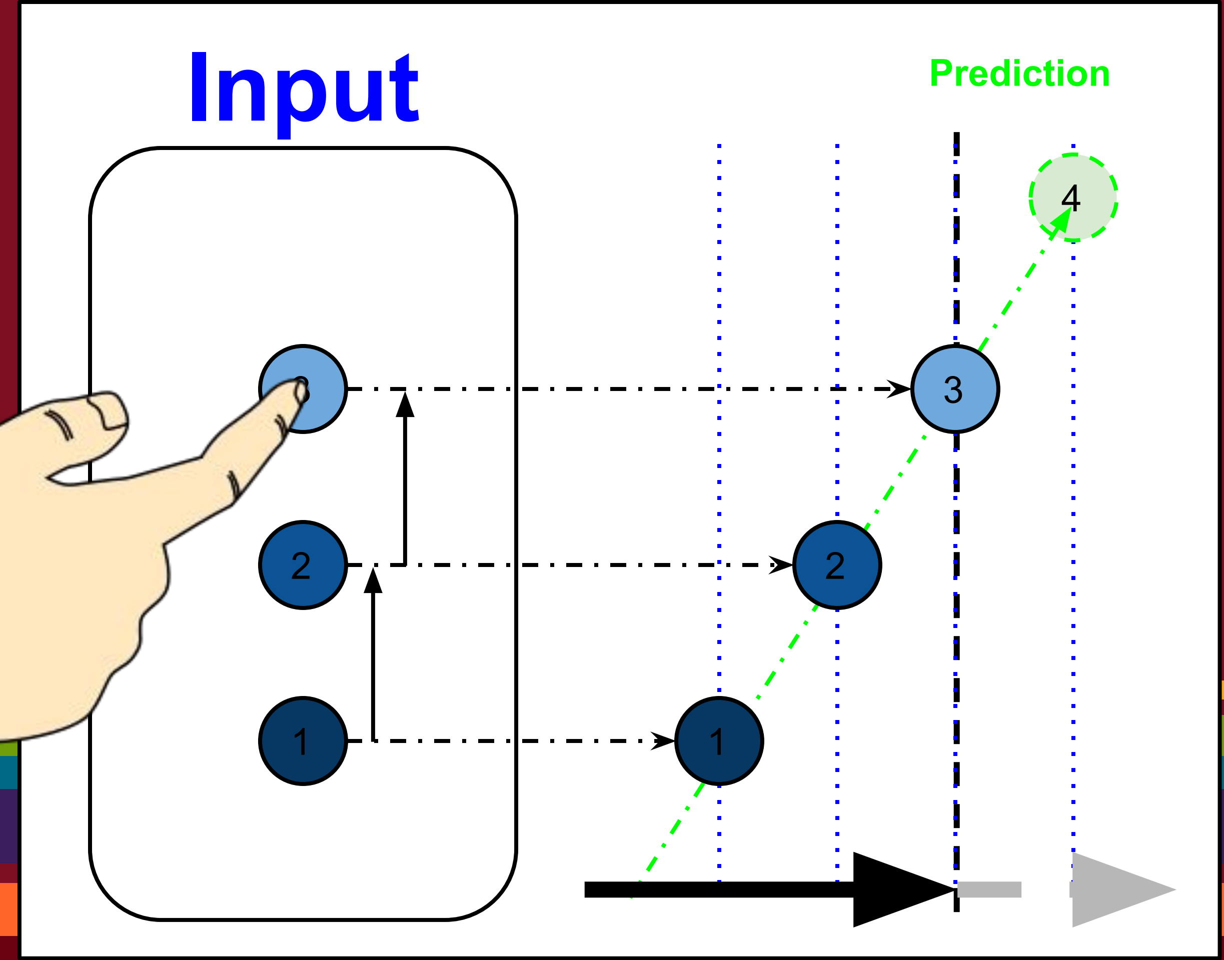
Input



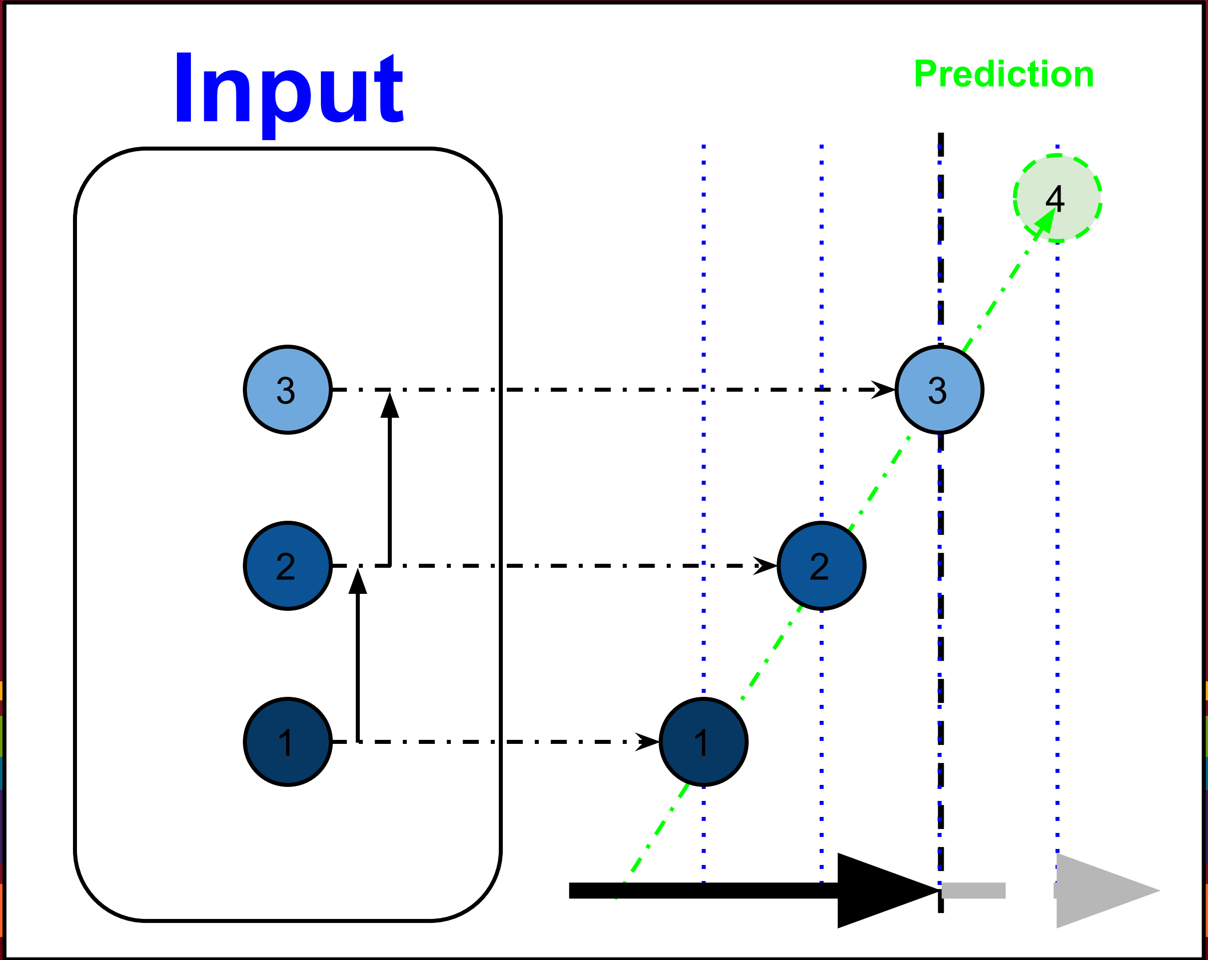
http://goo.gl/MHGWTc



http://goo.gl/MHGWTc



http://goo.gl/MHGWTc



http://goo.gl/MHGWTc