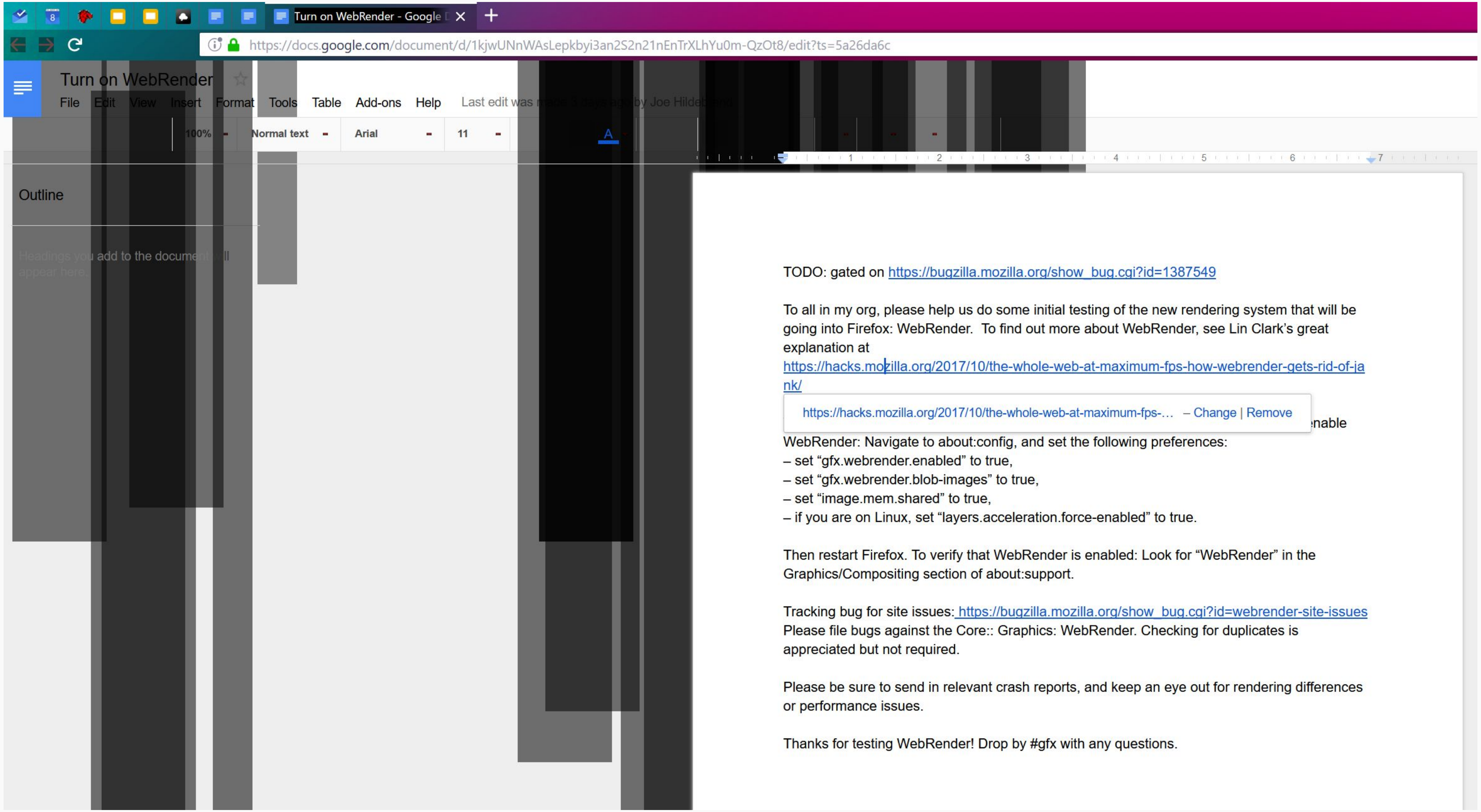
WebRender Capturing Infrastructure

Dzmitry Malyshau

Rust Toronto, 2nd July 2019


Turn on WebRender - Google D
https://docs.google.com/document/d/1kjwUNnWAsLepkbyi3an2S2n21nEnTrXLhYu0m-QzOt8/edit?ts=5a26da6c
Turn on WebRender
File Edit View Insert Format Tools Table Add-ons Help
100%
Normal text
Arial
11
Outline
Headings you add to the document will appear here.
https://hacks.mozilla.org/2017/10/the-whole-web-at-maximum-fps-… – Change | Remove

TODO: gated on https://bugzilla.mozilla.org/show_bug.cgi?id=1387549

To all in my org, please help us do some initial testing of the new rendering system that will be going into Firefox: WebRender. To find out more about WebRender, see Lin Clark's great explanation at https://hacks.mozilla.org/2017/10/the-whole-web-at-maximum-fps-how-webrender-gets-rid-of-jank/

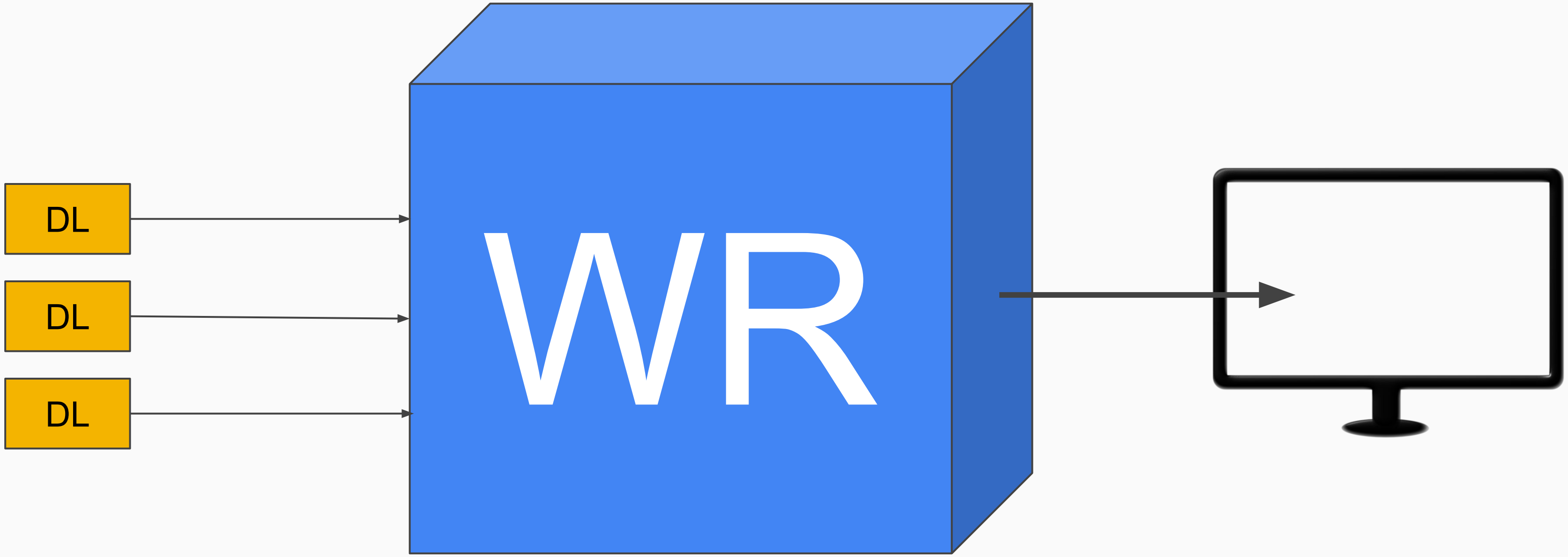
nable WebRender: Navigate to about:config, and set the following preferences:
– set "gfx.webrender.enabled" to true,
– set "gfx.webrender.blob-images" to true,
– set "image.mem.shared" to true,
– if you are on Linux, set "layers.acceleration.force-enabled" to true.

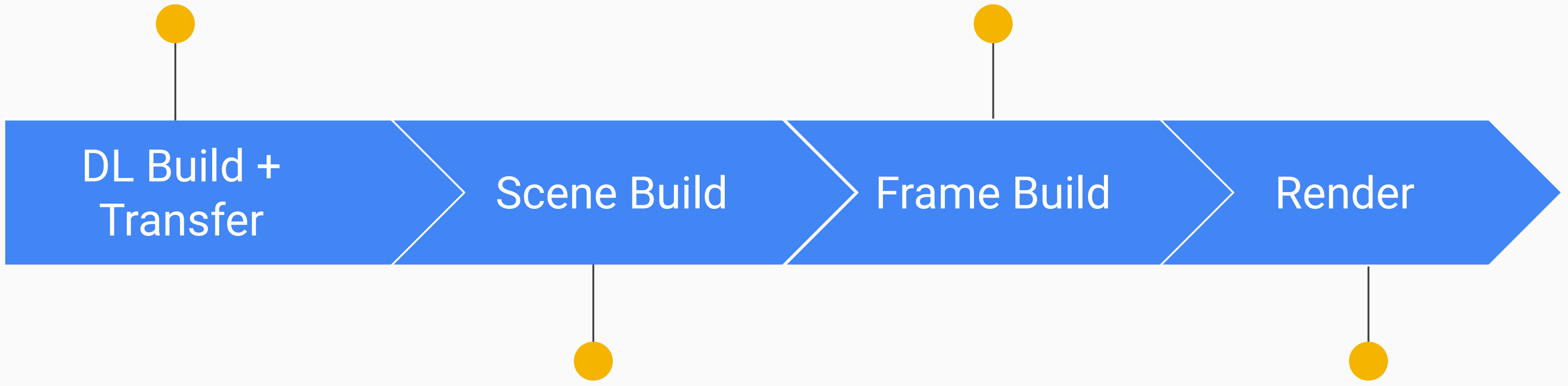
Then restart Firefox. To verify that WebRender is enabled: Look for "WebRender" in the Graphics/Compositing section of about:support.

Tracking bug for site issues: https://bugzilla.mozilla.org/show_bug.cgi?id=webrender-site-issues
Please file bugs against the Core:: Graphics: WebRender. Checking for duplicates is appreciated but not required.

Please be sure to send in relevant crash reports, and keep an eye out for rendering differences or performance issues.

Thanks for testing WebRender! Drop by #gfx with any questions.


DL
DL
DL
WR



Serialize user commands into data to be send to WR.

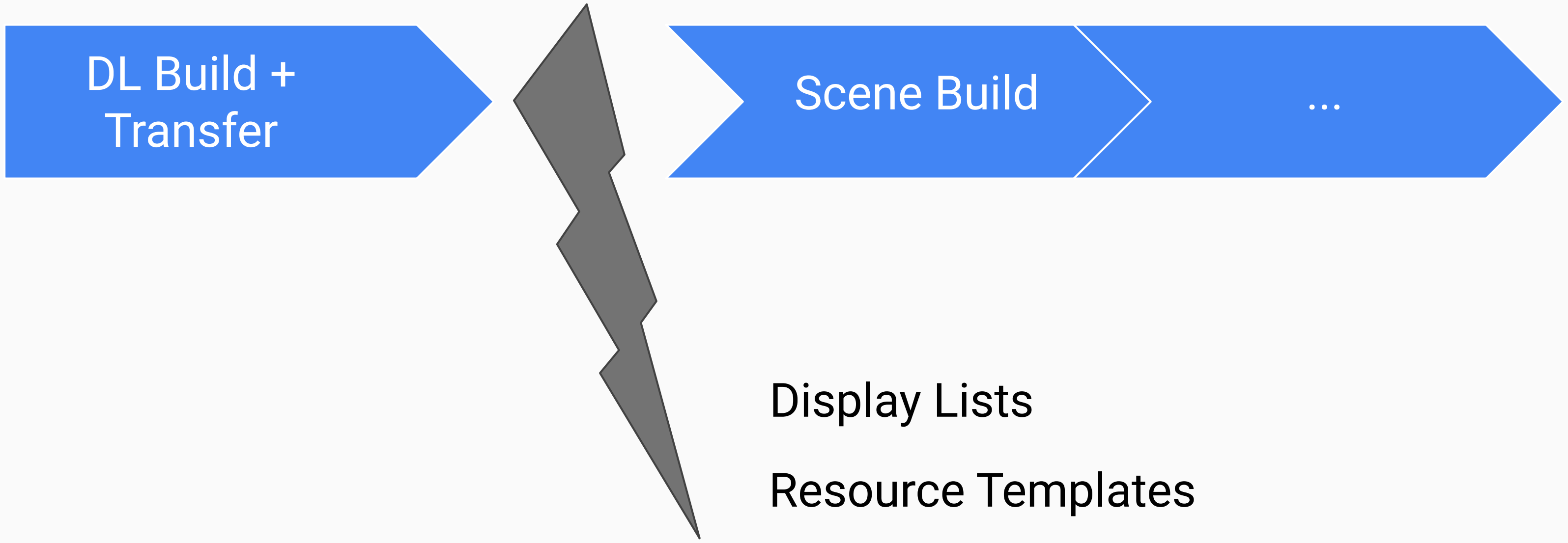
Build internal representation of the scene that we can easily scroll and animate.

Build batches and the render task tree, all ready to feed the driver.

Execute the frame commands and draw batches.

“The Cell” movie

WR Scene Capture

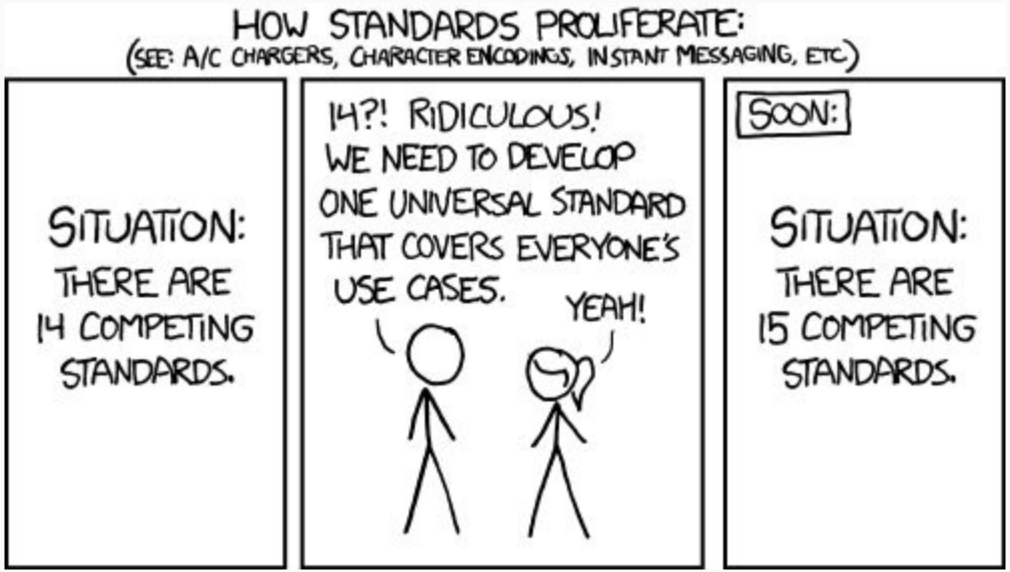


Display Lists

Resource Templates

scene-1-0.ron

```
root_pipeline_id: Some((1, 12)),
pipelines: {
    (1, 10): (
        pipeline_id: (1, 10),
        viewport_size: (2560, 1796),
        content_size: (2560, 1796),
        background_color: None,
        display_list: [
            PushStackingContext((
                origin: (0, 0),
                spatial_id: (1, (1, 10)),
                is_backface_visible: true,
                stacking_context: (
                    transform_style: Flat,
                    mix_blend_mode: Normal,
                    clip_id: Some(Clip(0, (1, 10))),
                    raster_space: Screen,
                    cache_tiles: true,
                ),
            )),// [0]
            Rectangle((
                common: (
                    clip_rect: ((0, 0), (2560, 1796)),
                    clip_id: Clip(0, (1, 10)),
                    spatial_id: (1, (1, 10)),
                    hit_info: None,
                    is_backface_visible: true,
                ),
                color: (
                    r: 0.9764706492424011,
                    g: 0.9764706492424011,
                    b: 0.9803922176361084,
                    a: 1,
                ),
            )),// [1]
            HitTest((
                common: (
                    clip_rect: ((0, 0), (2560, 1796)),
                    clip_id: Clip(0, (1, 10)),
                    spatial_id: (1, (1, 10)),
                    hit_info: Some((0, 1)),
                    is_backface_visible: true,
                ),
            )),// [2]
```


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

RON ?%!%

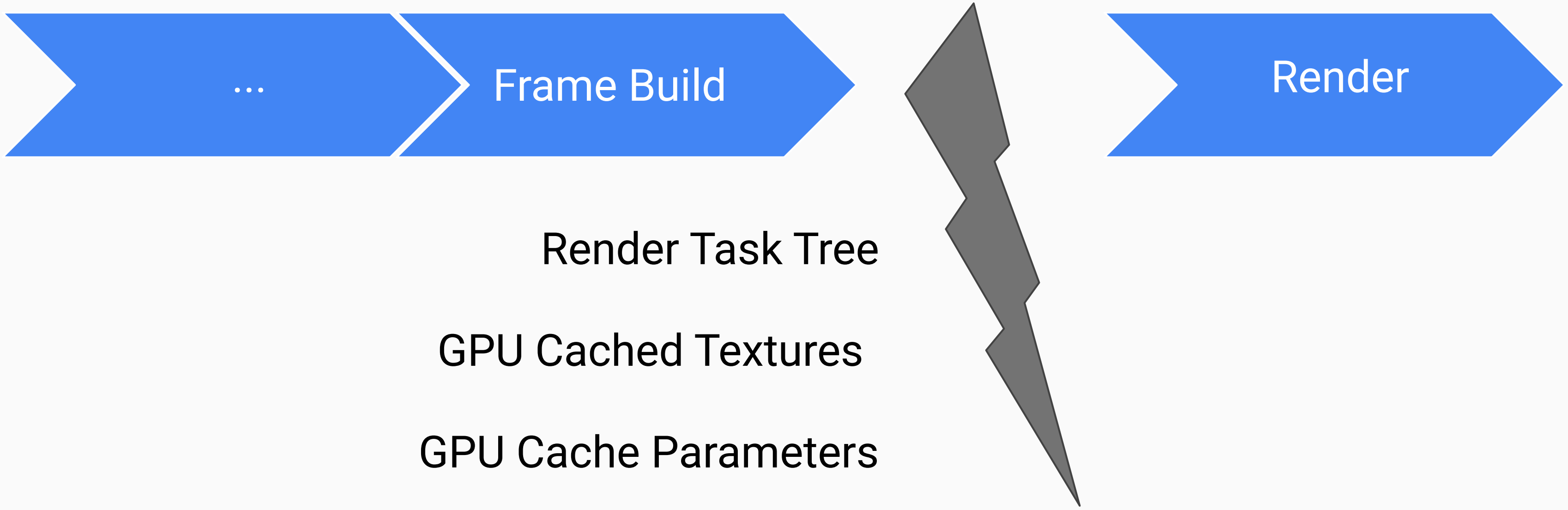
RON

A format we deserve
… but not necessarily need
https://github.com/ron-rs/ron

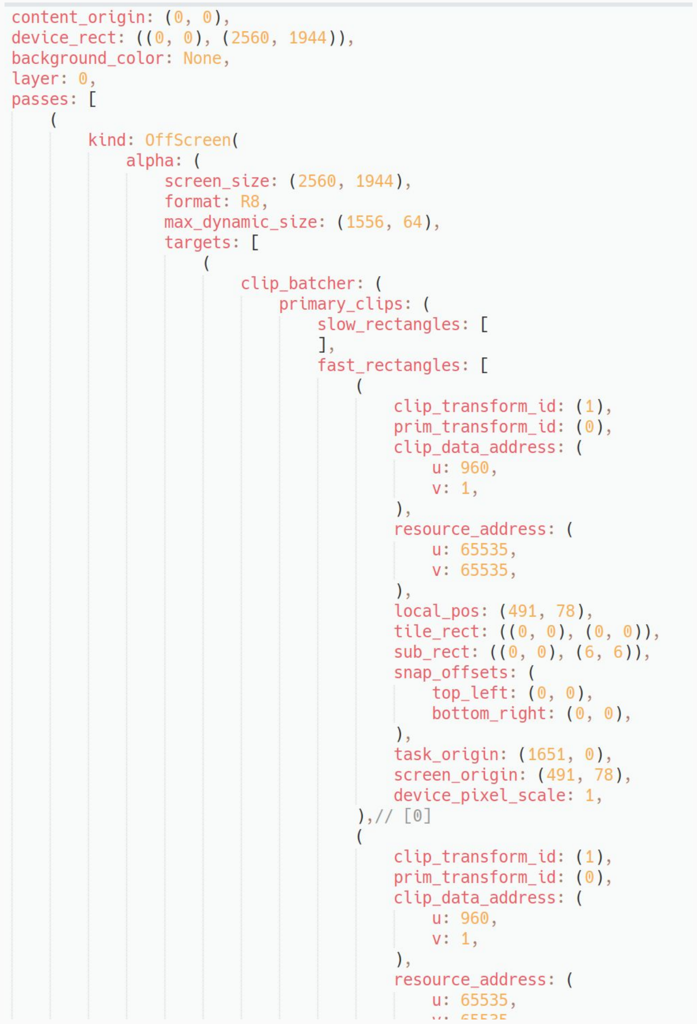
Features:

- Enums
- Typed maps
- Structs
- Tuples
- Indices
- Editor plugins!

WR Frame Capture

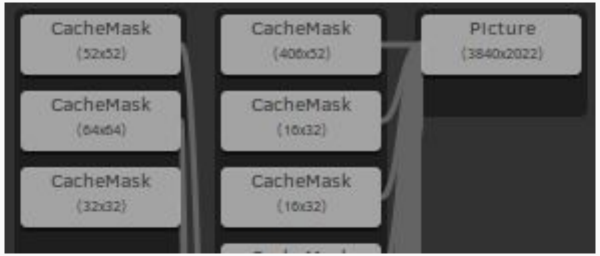


frame-1-0.ron



```
content_origin: (0, 0),
device_rect: ((0, 0), (2560, 1944)),
background_color: None,
layer: 0,
passes: [
    (
        kind: OffScreen(
            alpha: (
                screen_size: (2560, 1944),
                format: R8,
                max_dynamic_size: (1556, 64),
                targets: [
                    (
                        clip_batcher: (
                            primary_clips: (
                                slow_rectangles: [
                                ],
                                fast_rectangles: [
                                    (
                                        clip_transform_id: (1),
                                        prim_transform_id: (0),
                                        clip_data_address: (
                                            u: 960,
                                            v: 1,
                                        ),
                                        resource_address: (
                                            u: 65535,
                                            v: 65535,
                                        ),
                                        local_pos: (491, 78),
                                        tile_rect: ((0, 0), (0, 0)),
                                        sub_rect: ((0, 0), (6, 6)),
                                        snap_offsets: (
                                            top_left: (0, 0),
                                            bottom_right: (0, 0),
                                        ),
                                        task_origin: (1651, 0),
                                        screen_origin: (491, 78),
                                        device_pixel_scale: 1,
                                    ),// [0]
                                    (
                                        clip_transform_id: (1),
                                        prim_transform_id: (0),
                                        clip_data_address: (
                                            u: 960,
                                            v: 1,
                                        ),
                                        resource_address: (
                                            u: 65535,
```

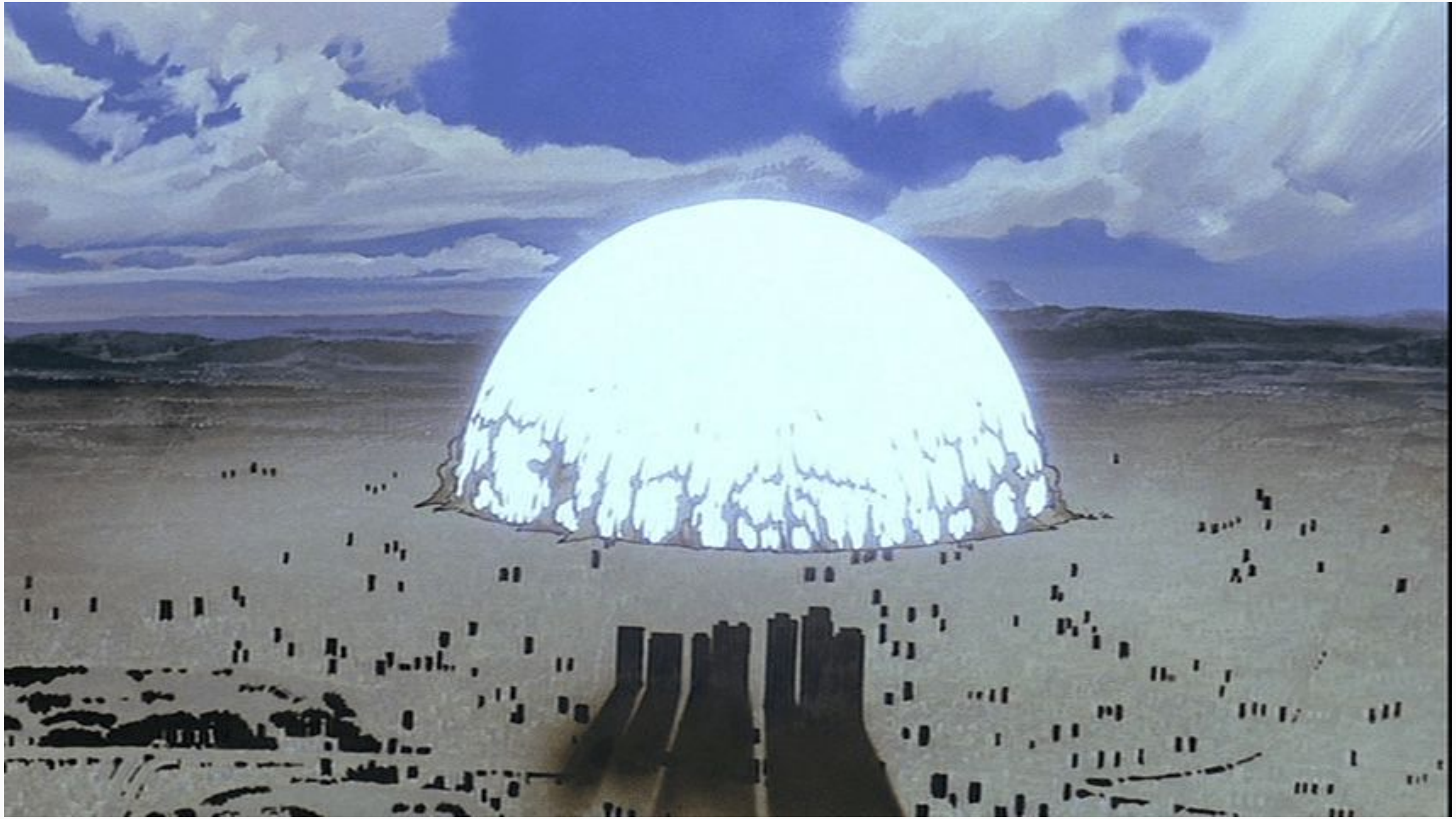
WR Captured Extras



- Spatial tree
- Interned primitive data
- Render task graph
- Picture tree
- Clip tree

Ctrl + Shift + 3

Capture Trigger



Replaying with Wrench

1. cd gfx/wr/wrench
2. cargo run -- load <capture_path>
3. (profit!)

See the original blog post for more info:

http://kvark.github.io/webrender/debug/ron/2018/01/23/wr-capture-infra.html

WR Capture Properties

- Readable
- Interactive
- Interchangeable
- Portable

Capturing Gotchas

… and advanced usage tricks

1. WebRender revision!

“wr.txt” = mozilla-central
74145209605580569286554e106415b061c1f393

File tweaks

2. Setting "*root_pipeline_id*" (in "*scene-1-0.ron*") to the content pipeline removes the UI.

3. Renaming "*frame-1-0.ron*" to anything forces WR to rebuild the frame on loading the capture.

Troubleshooting

3. Picture caching? 🚫

4. Frame builder configuration

   “*frame_config*” in "*backend.ron*"

5. Texture/gpu cache

   Reset by hitting “Y” in Wrench

Thank You!

With special thanks to **Serde** authors <3