

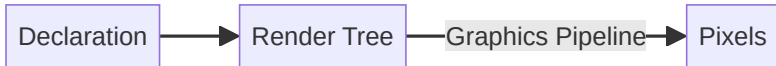
Immediate Mode Rendering Pattern

Who: Erik

Product: HeavyGoods.net

Lang: Typescript

Rendering (with a pipeline) 1

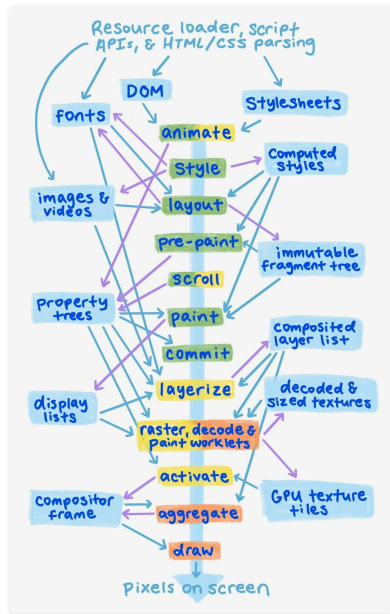


Rendering (with a pipeline) 2



Rendering (with a pipeline) 3





Oh No

- everything is sooo complex 🤖
- what if we're tasked creating the pixels ourselves?

Imagine we're the new Luddites:



to the rescue: "Immediate mode"



Immediate Mode Code

```
document.body.innerHTML = '<canvas style="background-color: white" />';
```

```
const canvas = document.getElementsByTagName("canvas")[0];  
const ctx = canvas.getContext("2d");
```

```
ctx.strokeRect(1, 1, 100, 20);  
ctx.fillText("OK", 30, 15);
```

```
canvas.onclick = ({ offsetX, offsetY }) => {  
  if (offsetX > 1 && offsetX < 100 && offsetY > 1 && offsetY < 20) {  
    console.log("click");  
  }  
};
```

Immediate Mode Code



State change

```
const state = { ok: true };
```

```
setInterval(  
  () => {  
    ctx.clearRect(0, 0, canvas.width, canvas.height);  
    ctx.strokeRect(1, 1, 100, 20);  
    ctx.fillText(state.ok ? "OK" : "NOT OK", 30, 15);  
  },  
  (1 / 60) * 1000, // 60Hz  
);
```

```
canvas.onclick = ({ offsetX, offsetY }) => {  
  if (offsetX > 1 && offsetX < 100 && offsetY > 1 && offsetY < 20) {  
    state.ok = !state.ok;  
  }  
};
```

State change

OK

Nesting

```
const state = { ok: [true, true, false] };
```

```
setInterval(  
  () => {  
    ctx.clearRect(0, 0, canvas.width, canvas.height);  
    for (const i of [0, 1, 2]) {  
      ctx.strokeRect(1, 1 + 25 * i, 100, 20);  
      ctx.fillText(state.ok[i] ? "OK" : "NOT OK", 30, 15 + 25 * i);  
    }  
  },  
  (1 / 60) * 1000, // 60Hz  
);
```

```
canvas.onclick = ({ offsetX, offsetY } => { ... })
```



Functional Programming

```
function button(env, { x, y, width, height, label, onclick }) {  
  env.ctx.strokeRect(x, y, width, height);  
  env.ctx.fillText(label, x + 30, y + 15);  
  
  env.onclick.push(({ offsetX, offsetY }) => {  
    if (  
      offsetX > x &&  
      offsetX < x + width &&  
      offsetY > y &&  
      offsetY < y + height  
    ) {  
      onclick();  
    }  
  }));  
}
```

FP - Container

```
function stackLayout(env, { children, height }) {  
  let y = 1;  
  
  for (let i = 0; i < children.length; i++) {  
    const c = children[i];  
  
    c(env, { y });  
  
    y += height;  
  }  
}
```

FP - Composition

```
function root(env) {
  stackLayout(env, {
    height: 30,
    children: [0, 1, 2].map(
      (i) =>
        (env, { y }) =>
          button(env, {
            x: 1,
            y,
            width: 100,
            height: 20,
            label: state.ok[i] ? `OK ${i}` : `NOT OK ${i}`,
            onclick: () => {
              state.ok[i] = !state.ok[i];
            },
          }),
    ),
  });
}
```

FP - render1

```
function render1(canvas, root) {  
  const onclick = [];  
  const ctx = canvas.getContext("2d");  
  
  ctx.clearRect(0, 0, canvas.width, canvas.height);  
  
  root({ ctx, onclick });  
  
  canvas.onclick = (ev) => onclick.forEach((c) => c(ev));  
}
```

FP - Loop

```
let state = {ok: [true, true, false]}

setInterval(
  () => {
    render1(canvas, root);
  },
  (1 / 60) * 1000, // 60Hz
);
```


FP 

OK 0

OK 1

OK 2

Why?



- no invisible steps
- render trees live inside the stack
- simple implementation (few lines of code)
- simple to use and debug (looks like react)
- extreme control over the output (pixel perfect)
- works on any system



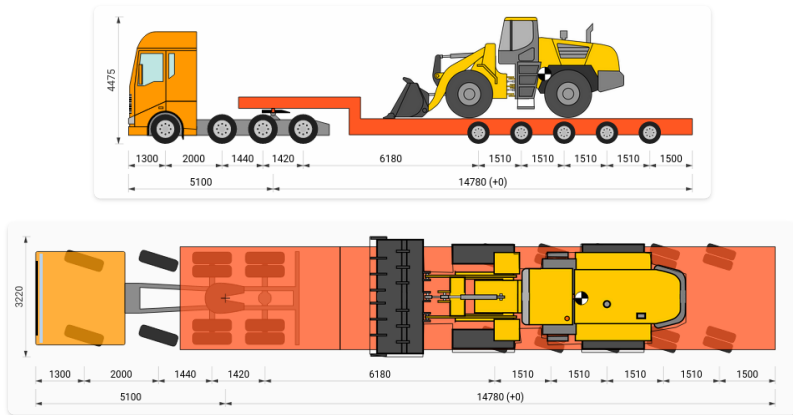
- computationally expensive (aka inefficient): renders everything all the time
- not standardized
- hard with layouts that require to know the size of some parts

Conclusion

- it can be done
- don't be afraid
- easily doable with basic FP- or OOP-skills (no endofunctors nor decorators required)

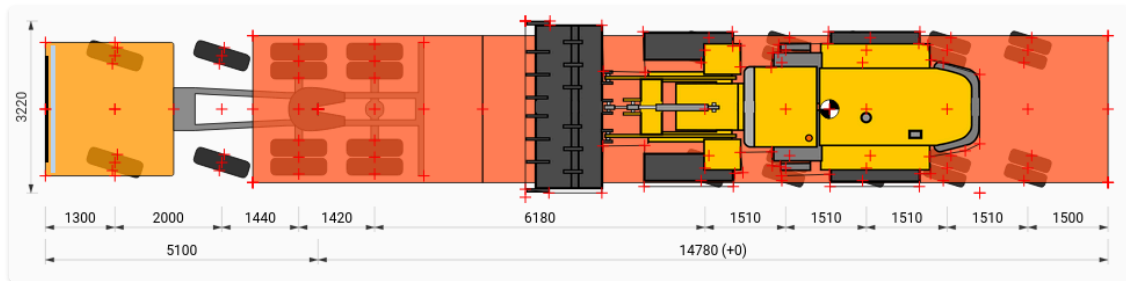
Anyone want to see it real life?

We use it in heavygoods.net to build simple vector-like vehicle depictions:

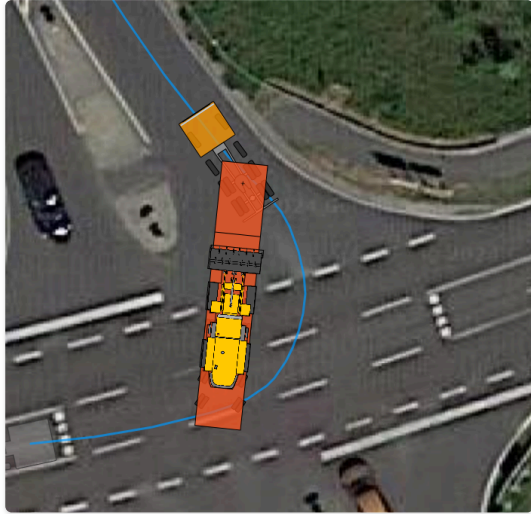


RL - debugging

With key points highlighted for debugging:



RL - full scene



RL - in a table

ATG16L1	ATG16L1-AS1	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS2	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS3	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS4	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS5	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS6	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS7	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS8	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS9	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS10	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS11	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS12	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS13	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS14	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS15	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS16	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS17	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS18	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS19	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS20	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS21	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS22	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS23	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS24	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS25	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS26	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS27	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS28	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS29	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS30	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS31	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS32	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS33	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS34	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS35	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS36	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS37	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS38	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS39	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS40	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS41	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS42	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS43	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS44	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS45	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS46	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS47	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS48	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS49	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS50	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS51	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS52	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS53	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS54	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS55	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS56	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS57	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS58	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS59	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS60	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS61	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS62	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS63	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS64	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS65	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS66	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS67	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS68	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS69	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS70	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS71	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS72	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS73	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS74	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS75	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS76	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS77	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS78	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS79	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS80	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS81	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS82	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS83	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS84	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS85	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS86	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS87	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS88	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS89	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS90	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS91	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS92	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS93	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS94	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS95	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS96	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS97	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS98	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS99	Transcript	100 bp	100	NC_010407.3
ATG16L1	ATG16L1-AS100	Transcript	100 bp	100	NC_010407.3

Implementation - Feats

Some tricks were necessary to make it work:

- DX: explicit 2D transformation matrix to group nested components
- DX: ctx is a facade to allow different render backends: svg, canvas, points, bounds
- DX: backends allow easy layouting
- DX: drawings have become models that contain truths (e.g. length of a vehicle is its drawing bounds 🤖)
- PERF: svg has a flat "virtual dom" for fast rerendering while dragging elements

Implementation - Actual Code

```
export function semiTrailerTop(  
  trailer: SemiTrailer | JointSemiTrailer,  
  vehicleState?: SimulationVehicleState,  
): Shape {  
  /* defines axles, virtualAxleShape, chassis */  
  
  return group({  
    children: [  
      ref({ key: "origin" }),  
      ref({ key: "chassisFrontJoint", pos: [0, 0] }),  
      axles,  
      virtualAxleShape,  
      chassis,  
    ],  
  });  
}
```

Thats all

Thanks!

Special Thanks: check out <https://sli.dev> - best tool so far to build slide decks with.

- built in diagrams with <https://mermaid.js.org/>
- built in code highlighting (& editing)
- my running examples are just vue components
- runs in github docs and exports to pdf