

# **interlace**

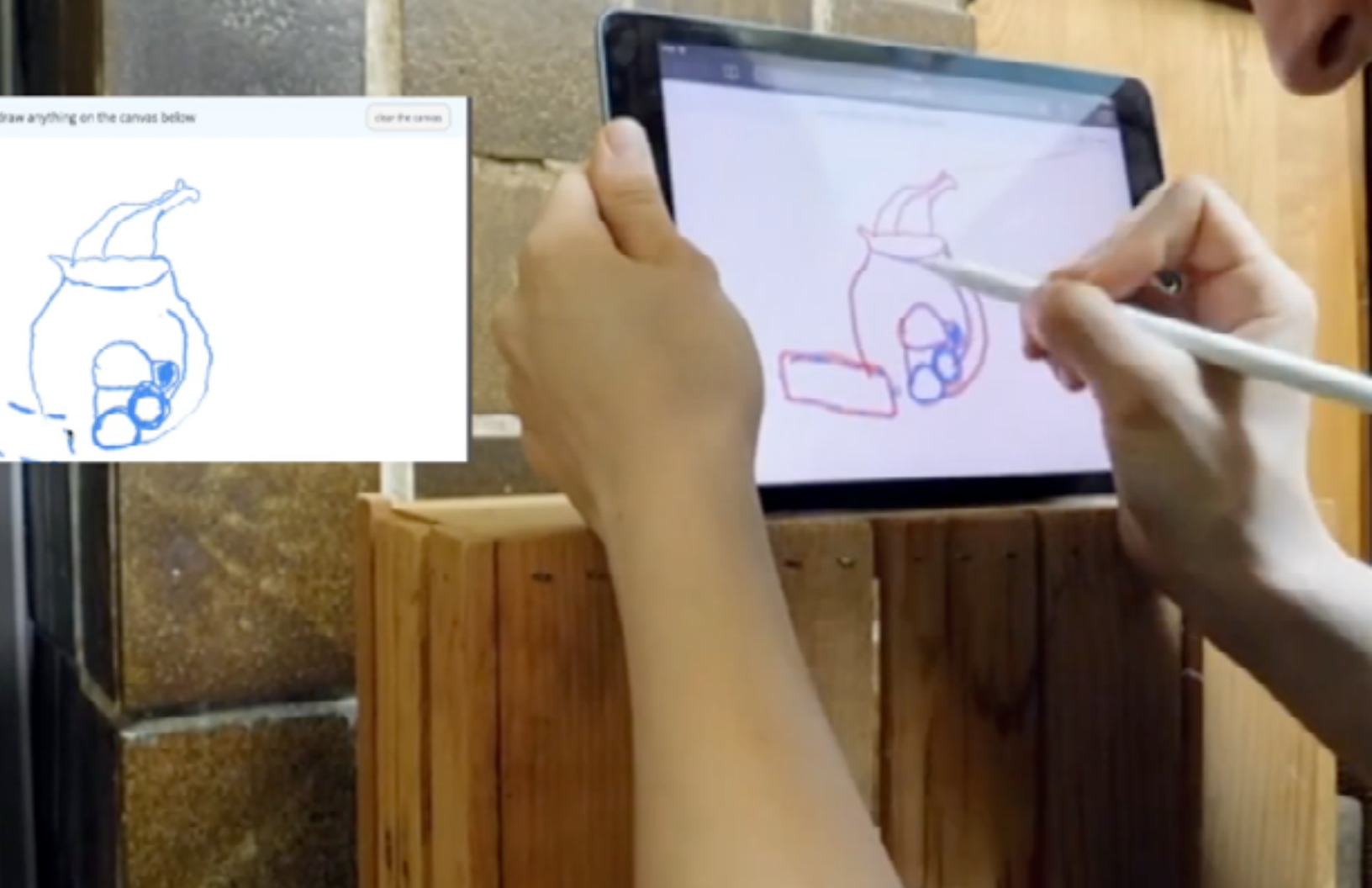
an interface for (de)augmented drawing

with the red strokes



# ***interlace***

a book of mediated drawings



***interlace*** is a software-mediated interface for drawing that both augments and de-augments the expressive powers of the artist. The novelty of the interface is derived from a simple implementation of a classic technique for artificial intelligence: *every line you draw is actually “implemented” by another human.*

This other human sees every line you draw and can choose to draw whatever they want on top. They can faithfully carry out your orders. Or they can disobey you, or they can assist you. You see only the lines the mediator chooses to make.

The image on the left shows the process of creating a co-drawing. On the left, the *illustrator* draws a still life. Those lines are transmitted in red to the *mediator*, who draws lines in blue. Their lines are transmitted in blue to the illustrator, who works further with these lines. The red traces in each co-drawing mark sites of dissonance and improvisation.

You can find a video of the interface in use at [hypotext.co/interlace](http://hypotext.co/interlace).

This book presents a series of drawings made by pairs of people drawing together as illustrator and mediator in ***interlace***, created in three sessions in Rochester, Spoleto, and New York.

In each session, the participants improvised new modalities of mediated drawing that I hadn't imagined before. For example, in Spoleto, participants invented the role of *dancer*, whose movements in front of a projector screen the illustrator and mediator sought to trace with lines on the dancer's body. In New York, participants invented the role of *storyteller*, whose words the illustrator and mediator sought to illustrate as soon as they entered the air.

kye  
Dec. 2018

performance at SCORES

*School of Making Thinking +  
Rochester Folk Arts Guild  
Middlesex, NY*

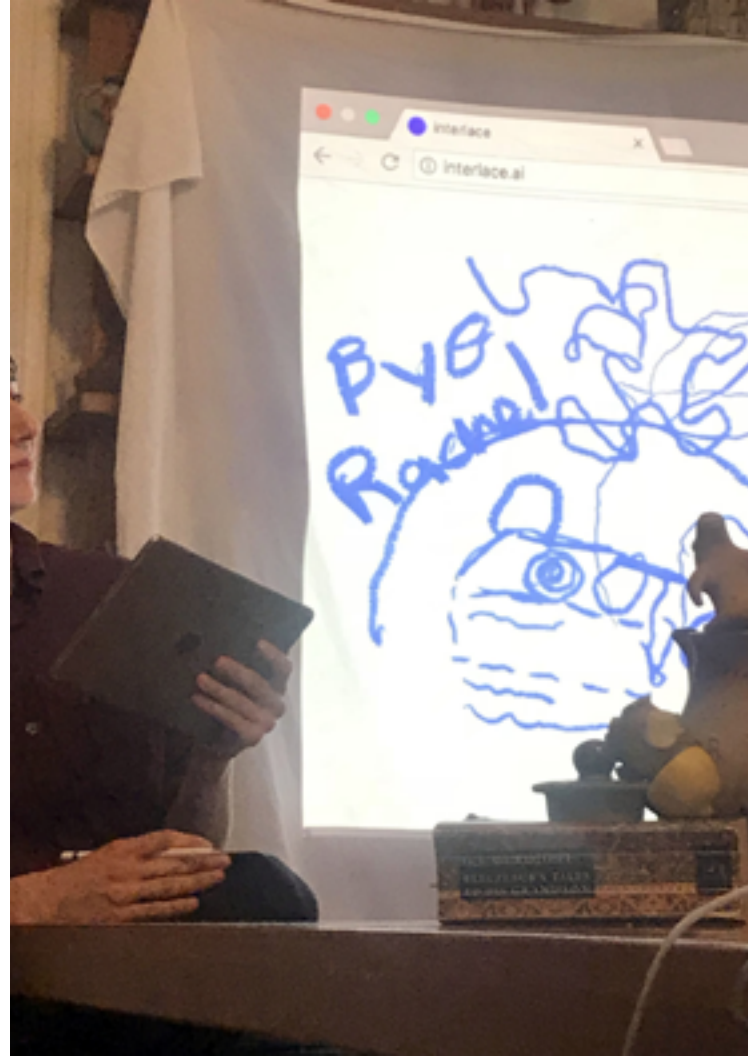
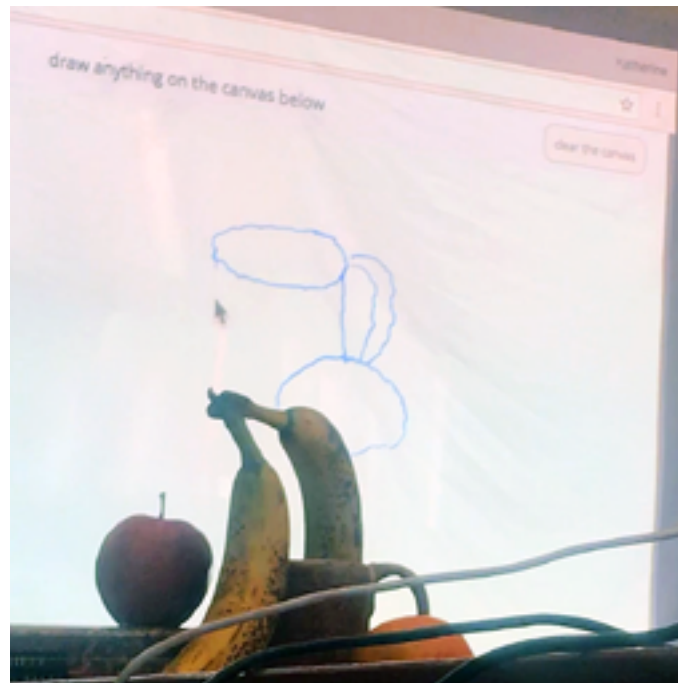


Georgia

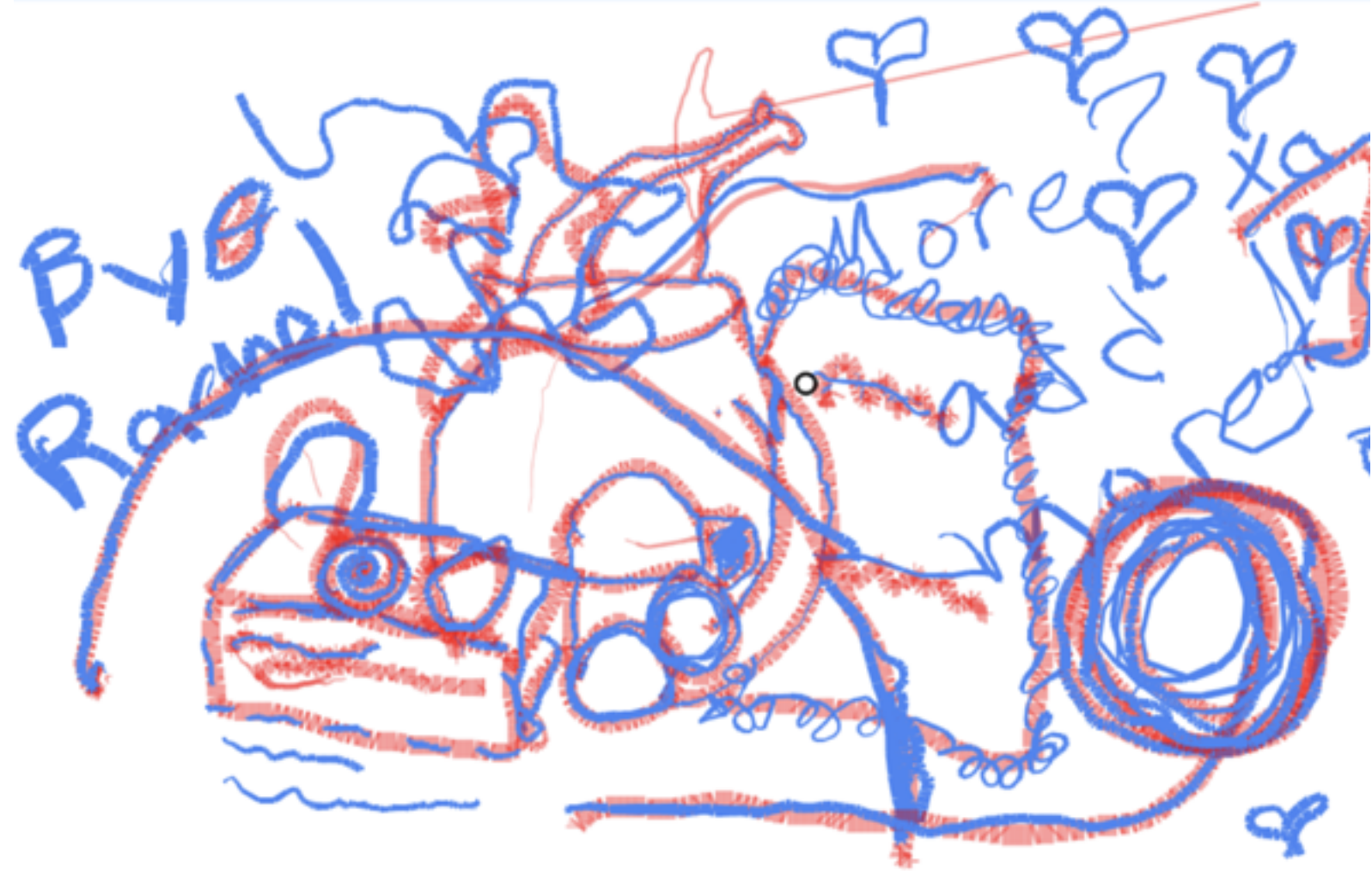


Georgia











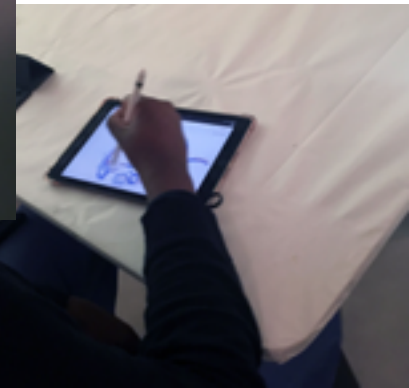
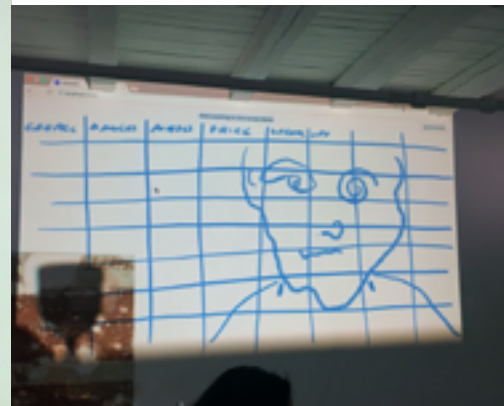
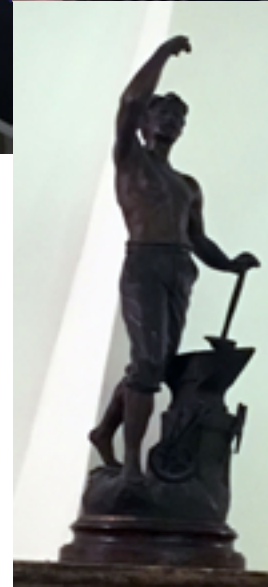
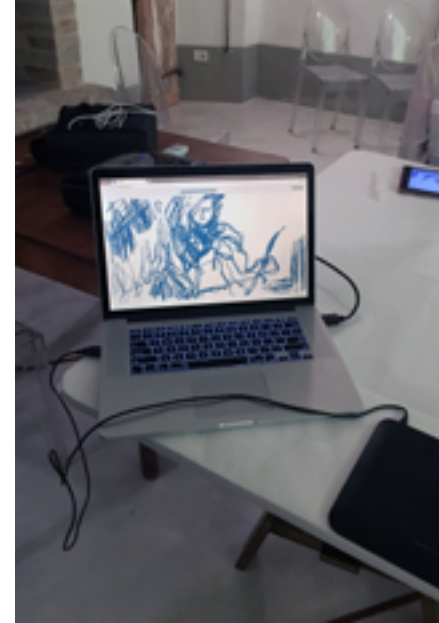
drawing sessions

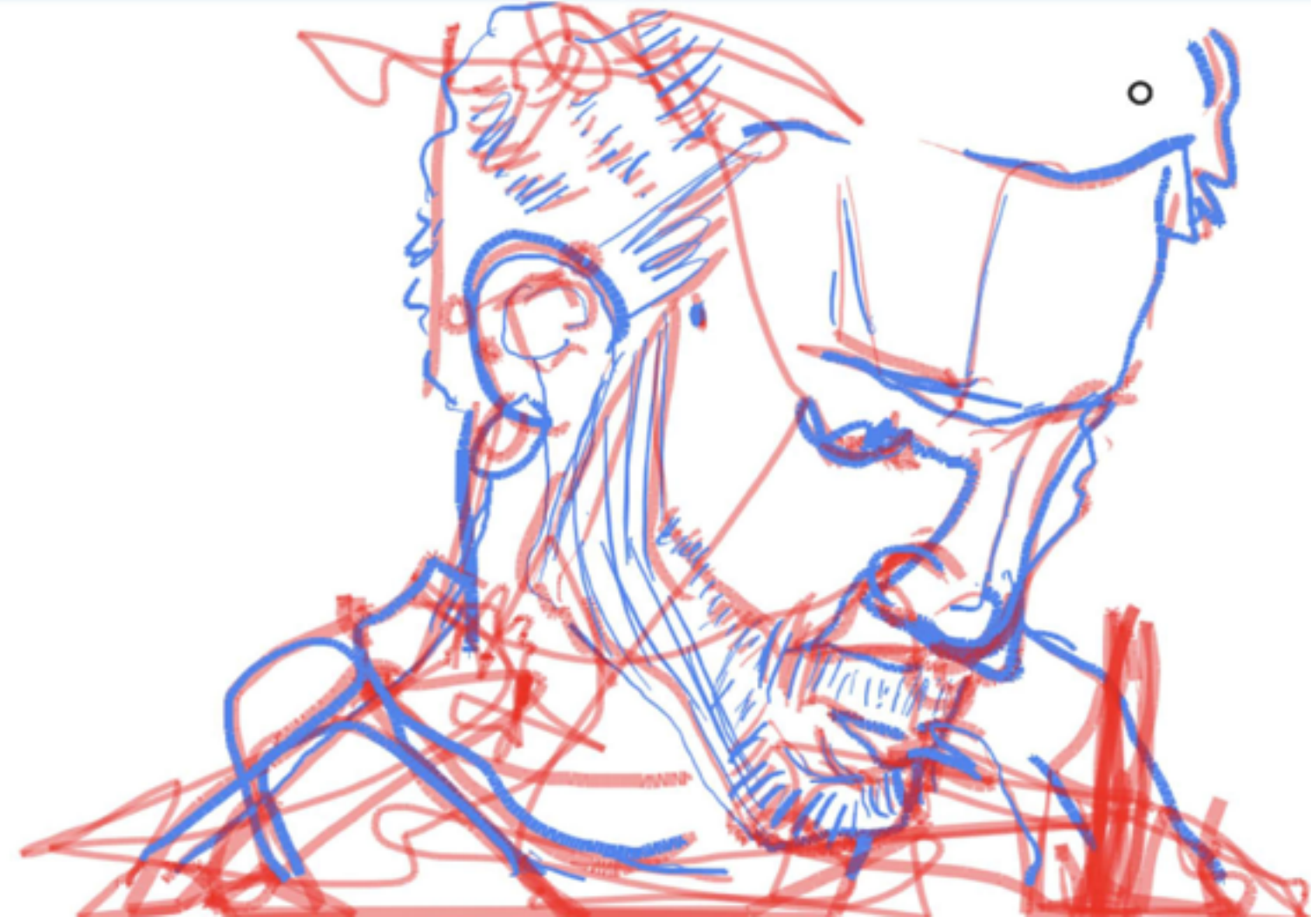
*Center for Arts Design + Social Research*  
*Spoleto, Italy*



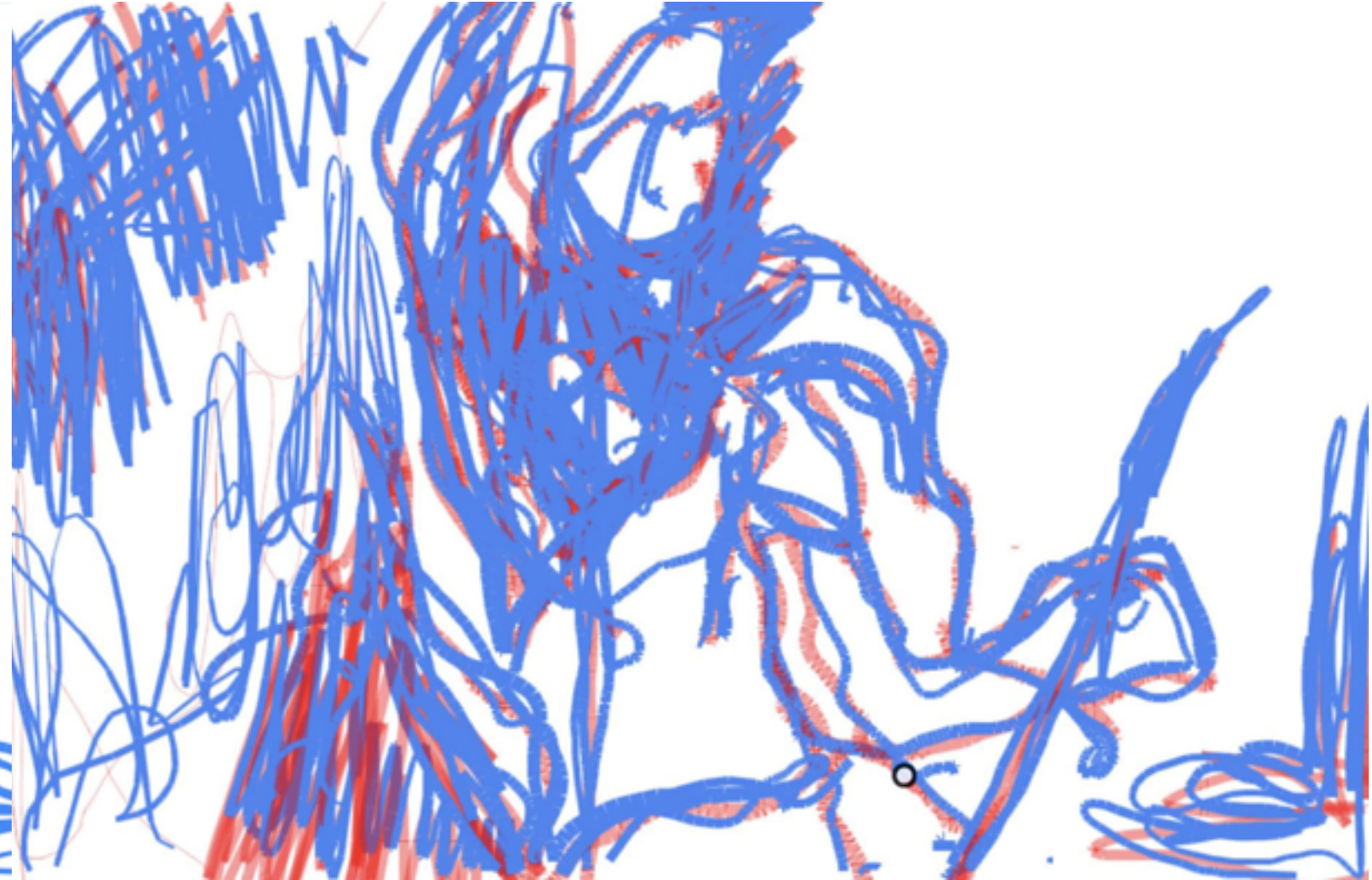
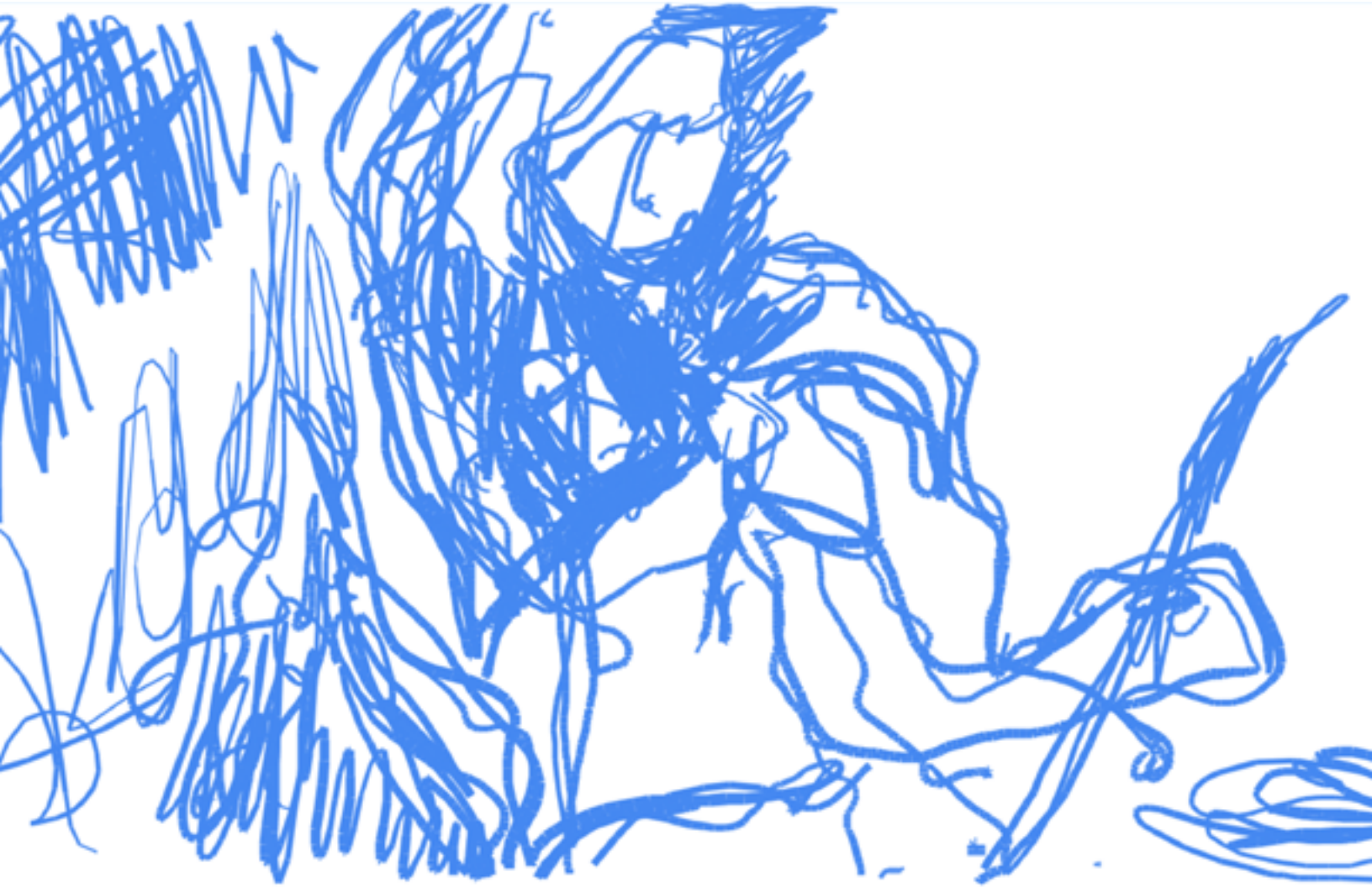


photo: RMO

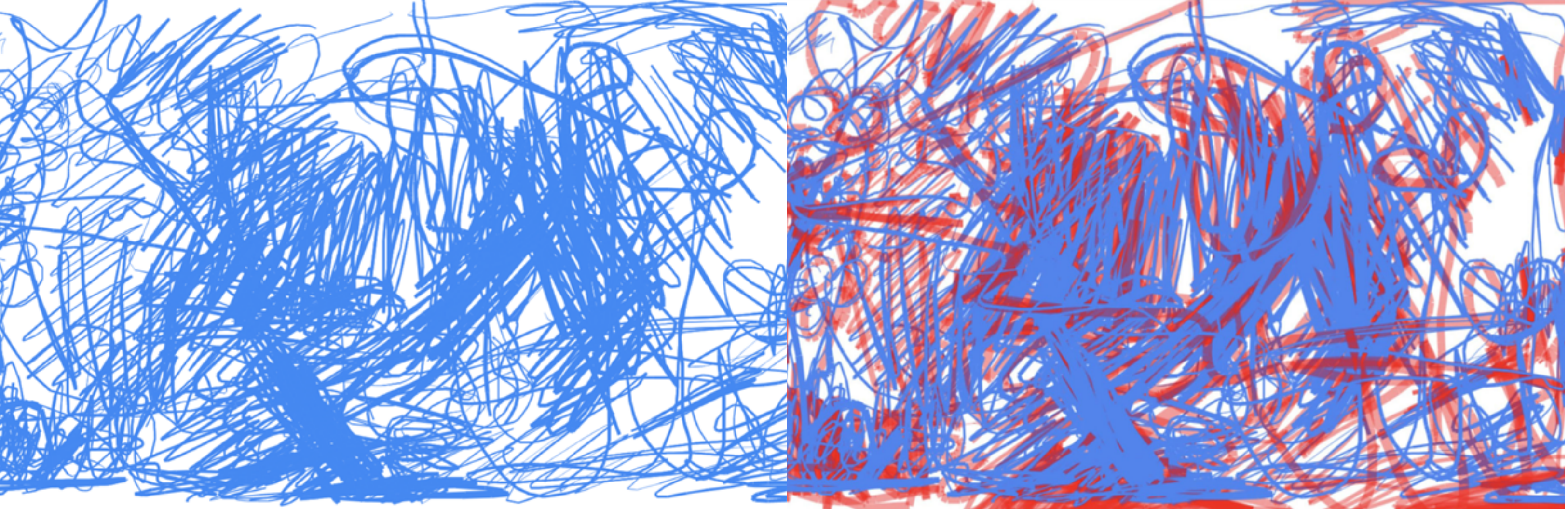








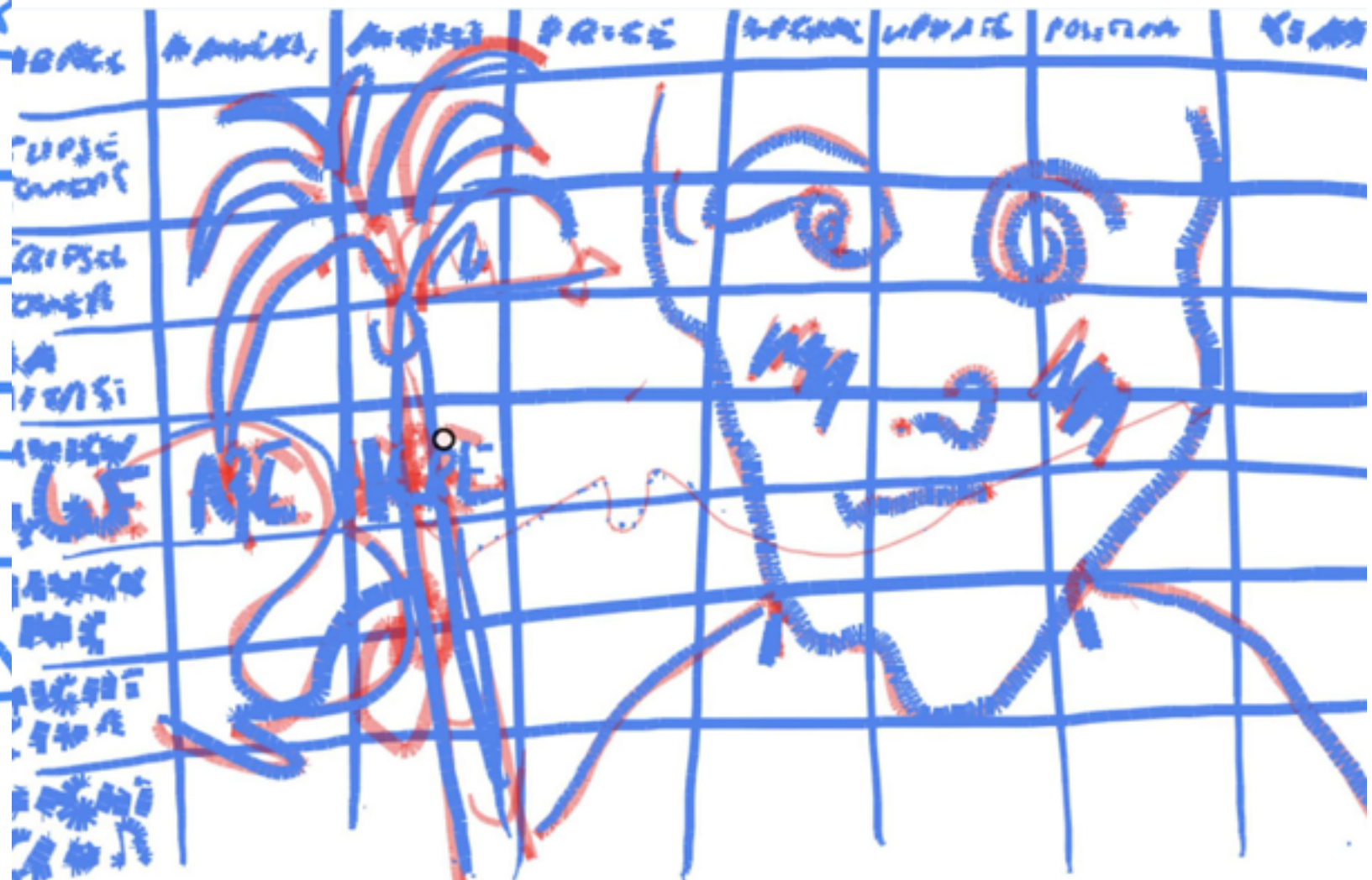


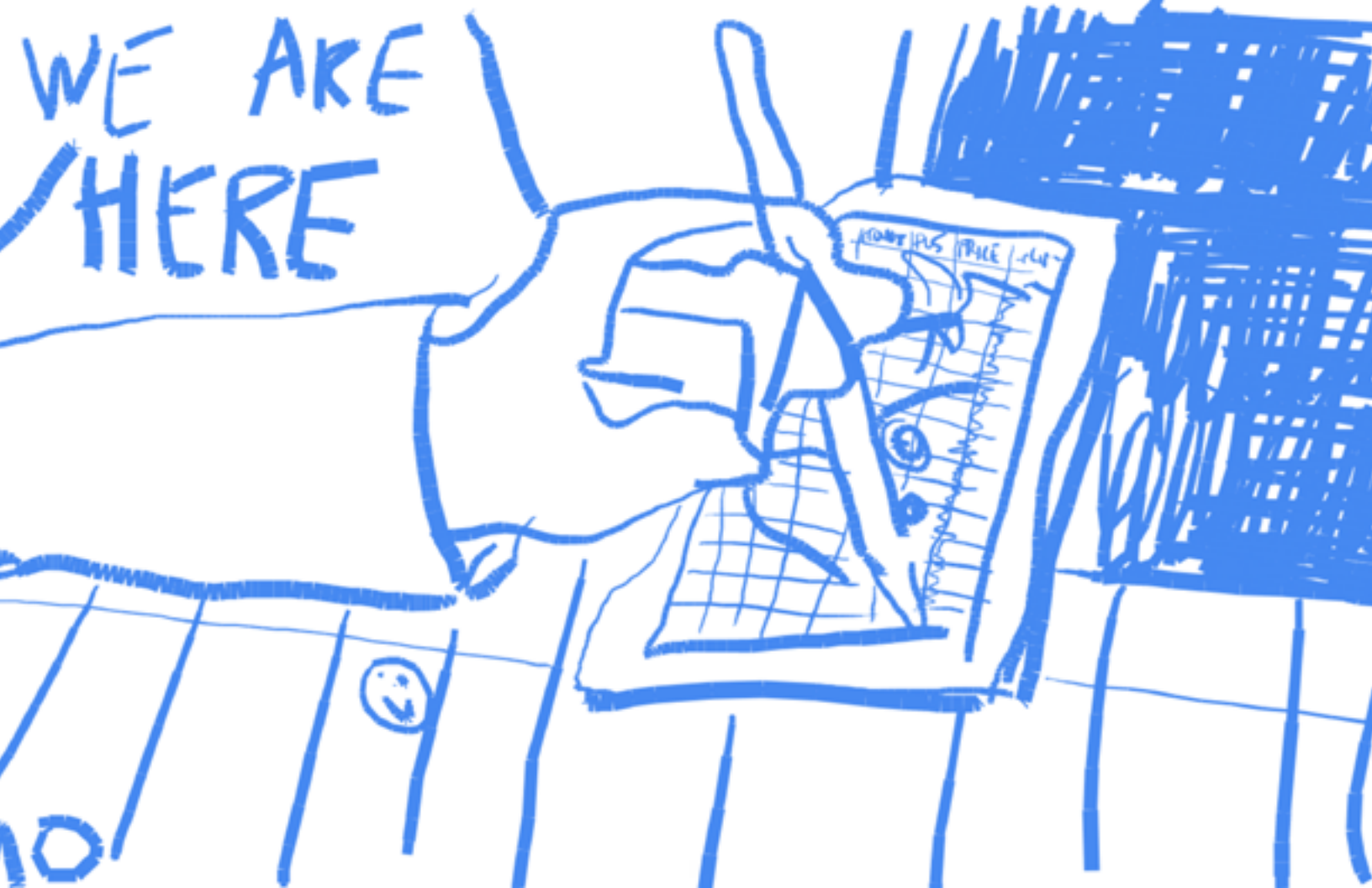




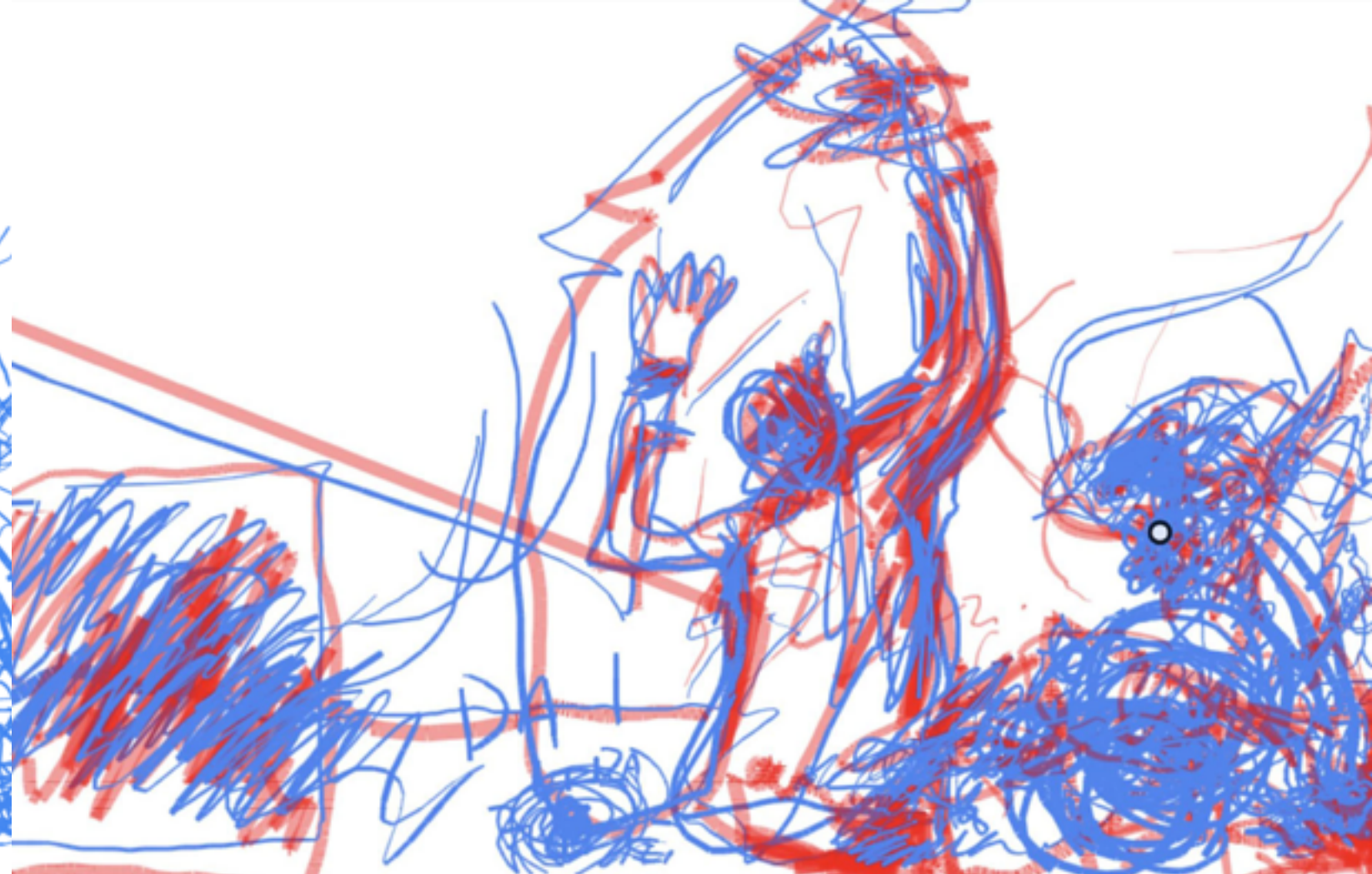
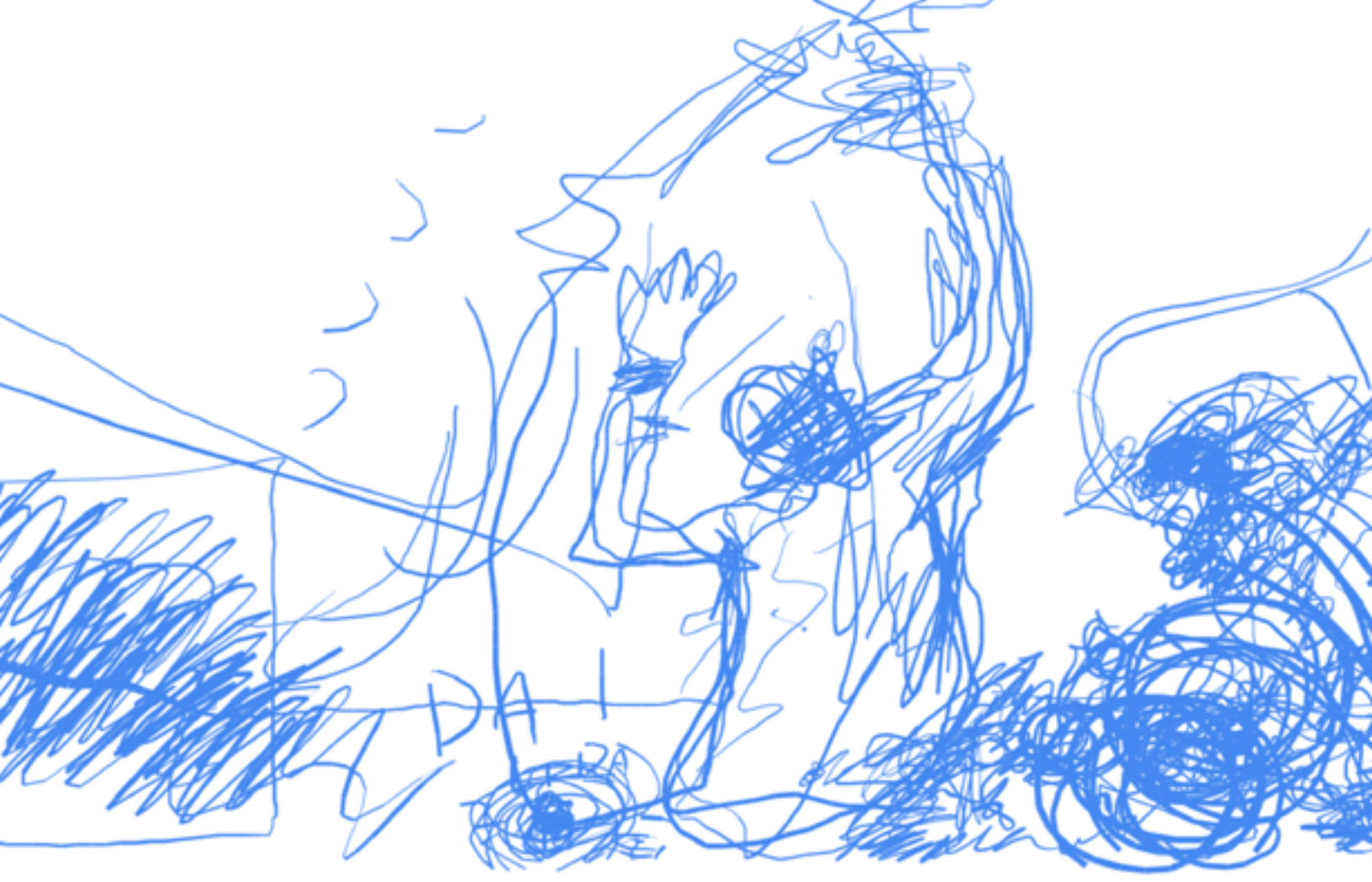




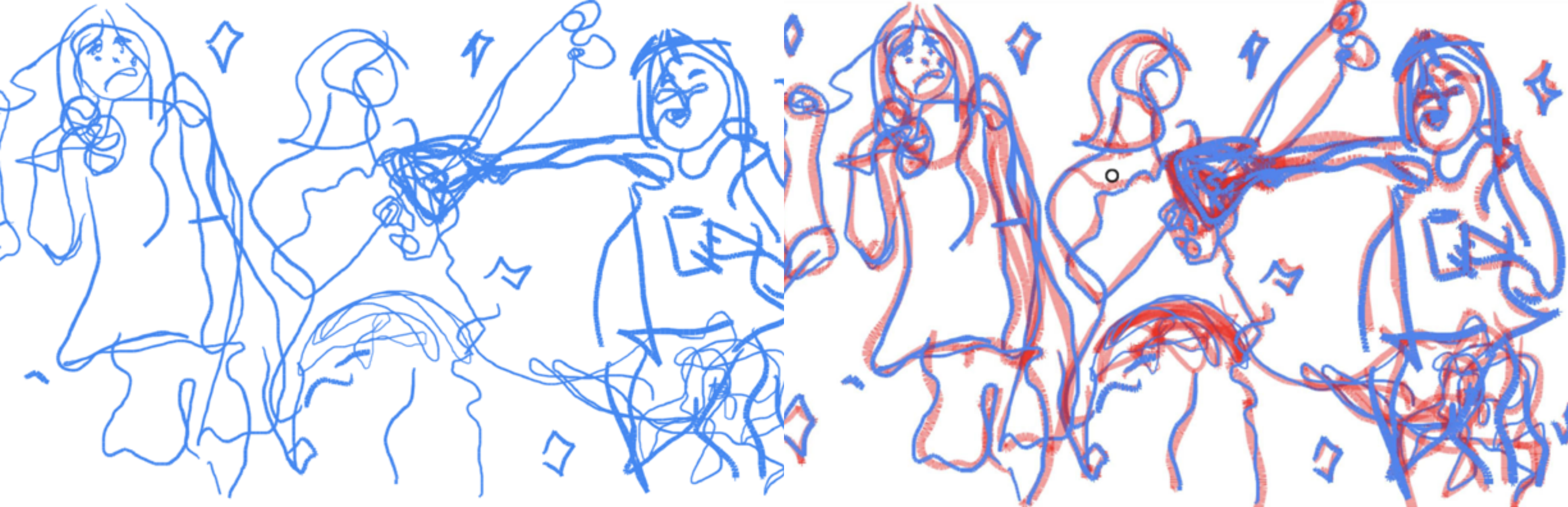


















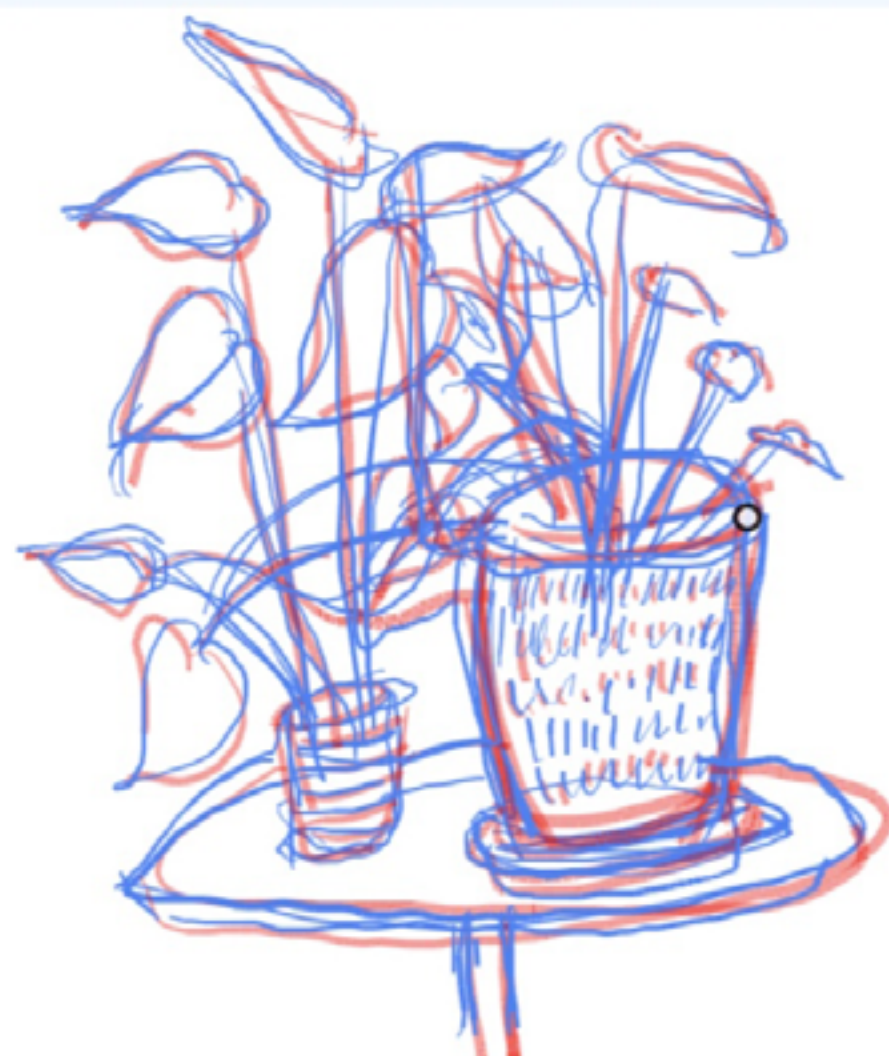


drawing sessions

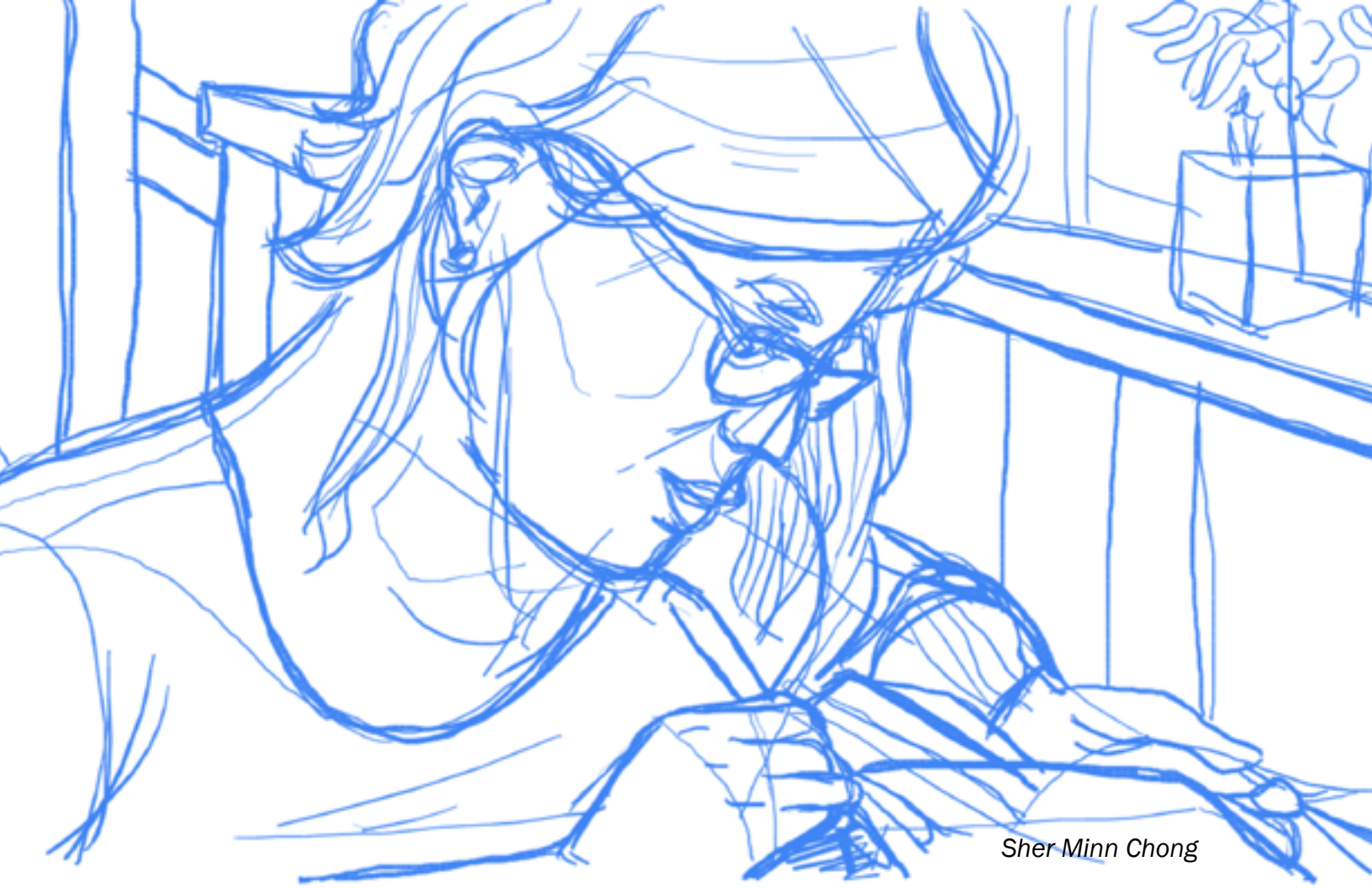
*TpT + Soft Surplus*  
*New York City*



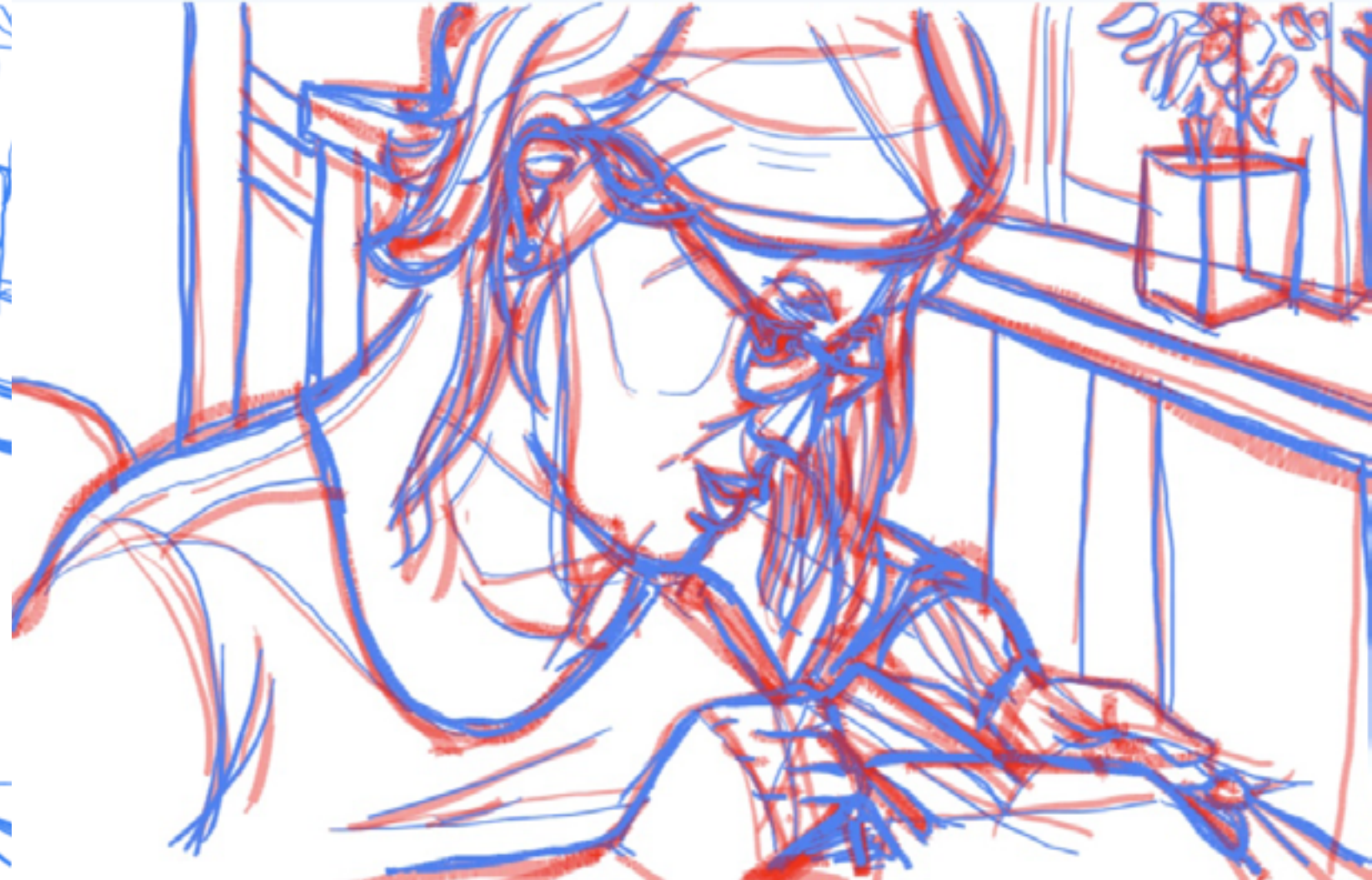
*Sher Minn Chong*





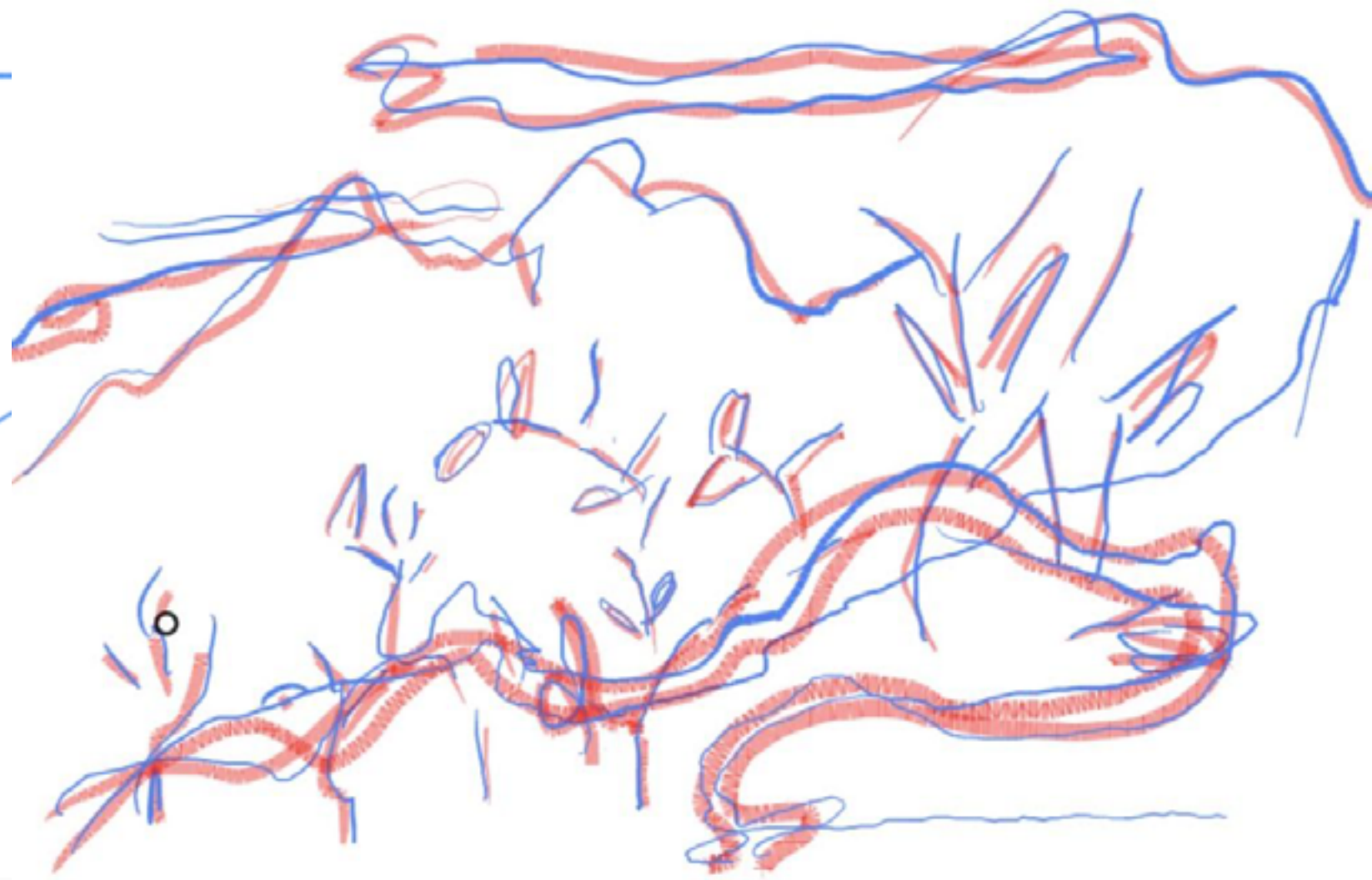


Sher Minn Chong

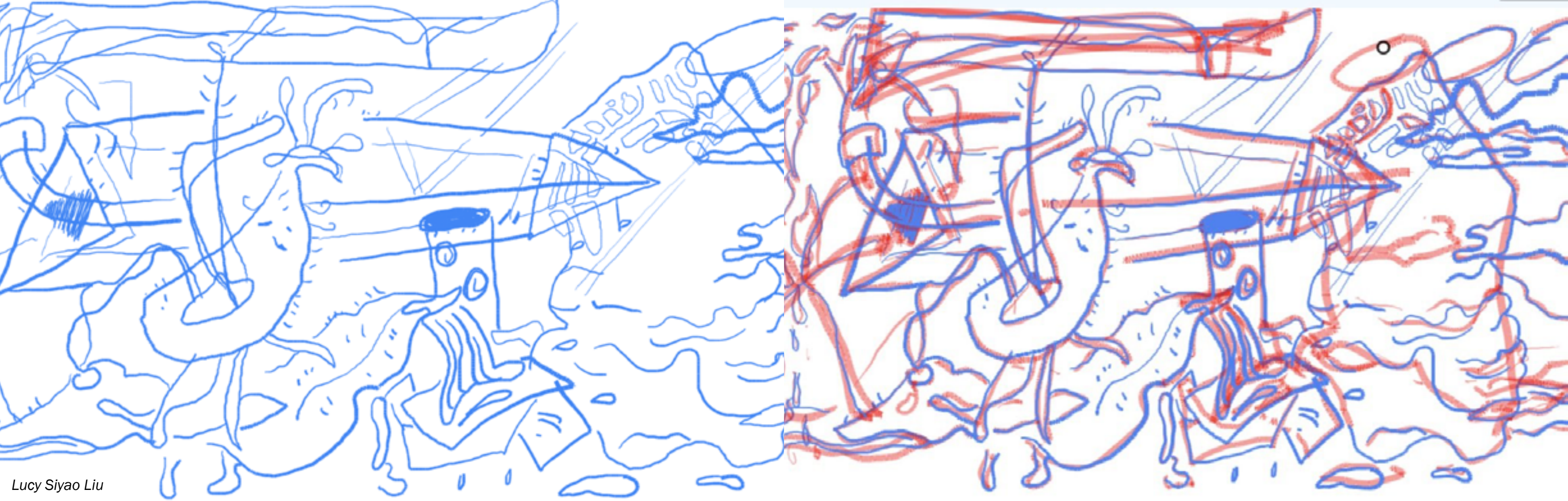




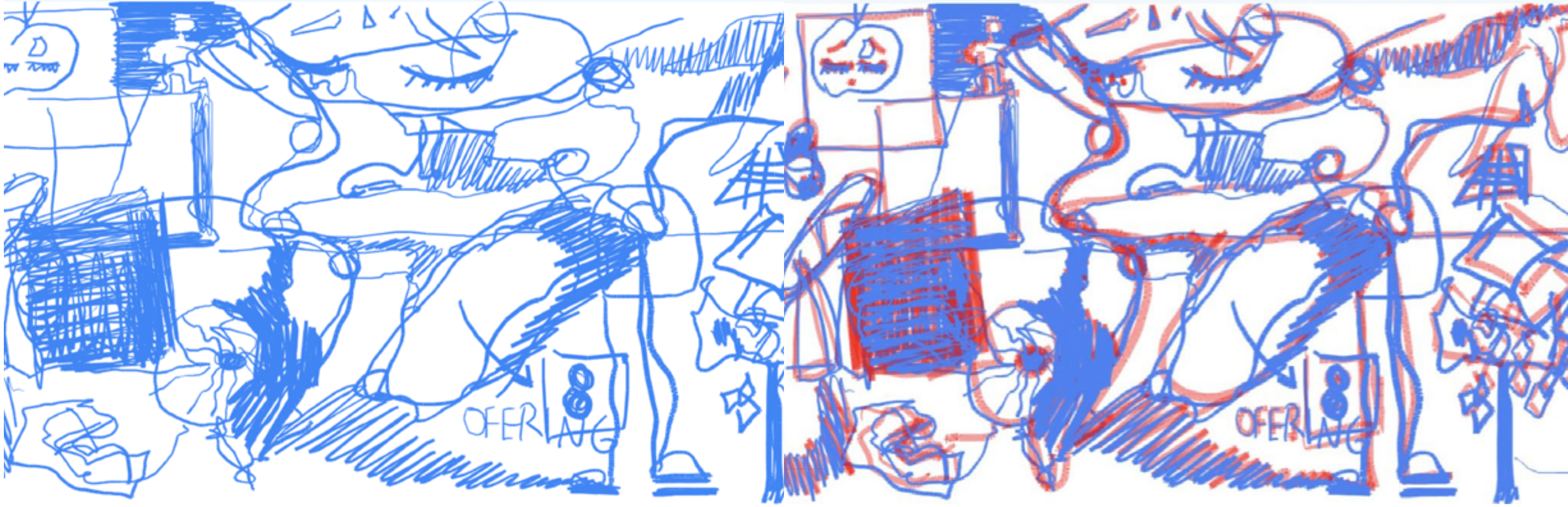
Lucy Siyao Liu











# make your own tools

```

client.handler for new (join) connections
let on('connection', function (socket) {
  var clients = new_clients + 1;
  console.log('Client connected %s',
    clients);
  console.log('New clients = ' + clients);
});

var init_game;

// give a player a role on join
// first to join is driver, second is mediator
if (! driver_taken) {
  console.log('Killerator joined');
  driver_taken = true;
  init_game = 1;
} else if (! mediator_taken) {
  console.log('Mediator joined');
  mediator_taken = true;
  init_game = 2;
} else if (! observer_client_taken) { // observer
  console.log('Observer of Killerator joined');
  observer_client_taken = true;
  init_game = 3;
} else { // all subsequent clients are observers of both
  console.log('Observer of both joined');
  init_game = 4;
}
}

// send 'player', { name: init_game };

// First send the history to the new client
for (var i in init_history) {
  socket.emit('new_game',
    { time: init_history[i].time, player_name: init_history[i].player_name, force: init_history[i].force });
}

// add handler for message type "draw_game",
socket.on('draw_game', function (data) {
  // add received time to history
  // TODO: use object copying function instead of this
  init_history.push({ time: data.time, player_name: data.player_name, force: data.force });
  // send time to all clients
  socket.emit('draw_game', { time: data.time, player_name: data.player_name, force: data.force });
});

socket.on('pl_player', function (data) {
  socket.emit('pl_player', {x: data.x, y: data.y});
});

socket.on('clear_game', function () {
  init_history = [];
  socket.emit('clear_game');
});
}

```

```

document.getElementById("showForm").onclick = function() {
    var state = {
        title: "Admin",
        type: "form",
        pos: [0, 0, 0, 0],
        pos_attr: "Admin",
        focus: false, focusArea:
    };

    var server_id = document.getElementById("server");

    // get canvas element and create context
    var canvas = document.getElementById("idCanvas_0");
    var ctx = canvas.getContext("2d");
    var context = canvas.getContext("2d");

    // var id = document.getElementById("idCanvas");
    // var id_target = id.idFromTarget; // includes position, size, color, name

    var width = window.innerWidth;
    var height = window.innerHeight; // id = id_height;
    var anchor = id.idFromID();

    // set canvas to full browser width/height
    // 5000 has more than area will enter if elements
    canvas.width = width;
    canvas.height = height;

    // register mouse event handlers

    // browser id is more documents arguments is the
    document.onmousedown = function(e) {
        // trigger id is 0
        // console.log("id mouse detected and browser id");

        // note here server id is id target plus offset screen size
        // 5000 offset not
        var mouseX = (context.idFromID(e));
        var mouseY = (e.pageY > 0 ? width : e.pageY < 0 ? height);

        context.fillStyle = "black";
        context.fillRect(
    );
    }

    document.onmouseover = function(e) { // console.log("mouse over"); //
        mouse.click = true; }

    document.onmouseout = function(e) { // console.log("mouse out"); //
        mouse.click = false;
        mouse.hover = false;
        mouse_pos_attr = false; }

    document.onmousemove = function(e) {
        var mouseX = canvas.idFromID(e);
        // function mouse position for range 0-8 = 0-8
        // console.log("mouse over", e.clientX, e.clientY, mouseX);
        mouse_pos.x = mouseX < 0 ? width;
        mouse_pos.y = mouseY < 0 ? height;
    }
}

```

[illegible][illegible]

```
// show P1's number location to P1, observers
watch.on('P1_number', function (data) {
  if (c1ayer != 1) {
    // console.log("P1 number received", data);
    // document
    var width = (x_data * 2 * width) / (y_data * 2 * height);
    // position on whole page, not just canvas
    var xPix = offsetCanvas * width;
  }

  var canvasWidth = 100;
  canvas_el.style.left = (x_data * (CANVAS_WIDTH / 2) + "px");
  canvas_el.style.top = (y_data * (CANVAS_HEIGHT / 2) + "px");
}

// =====
clear_canvas_listener("clear", function () {
  watch.off("clear_canvas");
});

// =====
watch.on('clear_canvas', function () {
  clearCanvas();
});

// =====
watch.on('player', function (data) {
  // only set on the first time
  if (c1ayer == 1) {
    c1ayer = data.id;

    console.log("c1ayer", c1ayer);

    // show cursor to all c1ayers except 00
    if (c1ayer != 10) {
      cursor_el.style.display = "block";
    }

    // show interaction text
    if (c1ayer == 10) {
      document.getElementById("P1").style.display = "block";
    } else if (c1ayer == 3) {
      document.getElementById("P3").style.display = "block";
    } else if (c1ayer == 11) {
      document.getElementById("P11").style.display = "block";
    } // don't let observers clear the canvas
    clear_canvas_listener("clear");
  } else { // all
    document.getElementById("P1").style.display = "block";
    // don't let observers clear the canvas
    clear_canvas_listener("clear");
  }
}

}
```

```

// main: main test run
// main loop, running every time
function mainLoop() {
  // check if the user is loading
  if (mainClick && mainMenu) {
    // console.log('user is loading')
    // console.log(main)

    if (mainMenu.open) {
      // player == 0 || player == 1 ?
      // add fire to the array
      // don't let observers draw
      mainMenu.drawFire =
        {
          // Line: (mainMenu, mainMenu,
            player, fire,
            fire: mainMenu
          }
        }
    }
  }

  // wait out for object reference to value (see screenshot)
  mainMenu.open = 0 < 1 ? mainMenu.open : 1 : mainMenu.open = 1;
  mainMenu.fire = 1;

  window.requestAnimationFrame(mainLoop);
}

window.requestAnimationFrame(mainLoop);

```

kye for SVN SYSTEMS  
\_\_ of \_\_ for the New York Tech Zine Fair  
Dec. 2018





