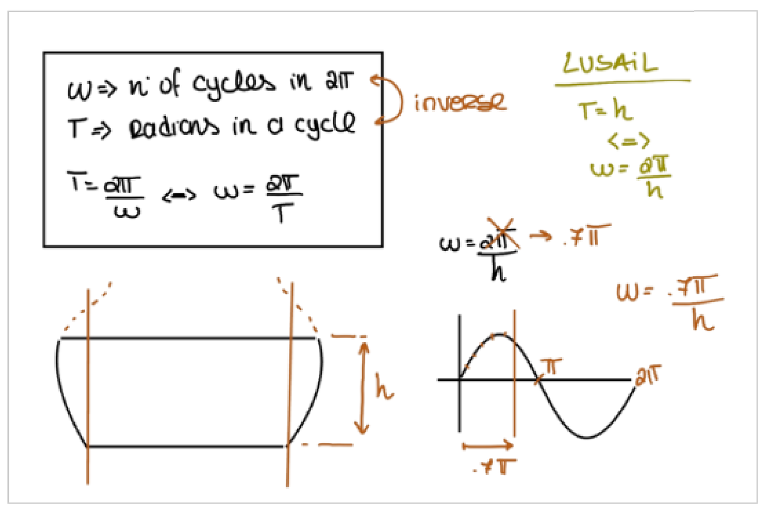
Lusail Stadium, Qatar

Algorithmic Design Project Adaptation | by Renata Castelo Branco

Original Architecture Project

The Lusail Stadium is the largest in Qatar and was designed by Foster + Partners for the 2022 FIFA World Cup. In a creative crossing of heritage-based inspirations, which included traditional arabe bowls and vessels, as well as Islamic lanterns, the building takes the shape of a golden cup with triangular perforations on the façade. The matrix of perforations mirrors the inner truss structure supporting the façade and the openings dictate the amount of natural light flowing into the galleries. In order to match the sinuous movement of the cup, the roof structure is shaped like a pringle and is composed of plastic membranes distributed in a radial pattern.



2 separate sinusoids

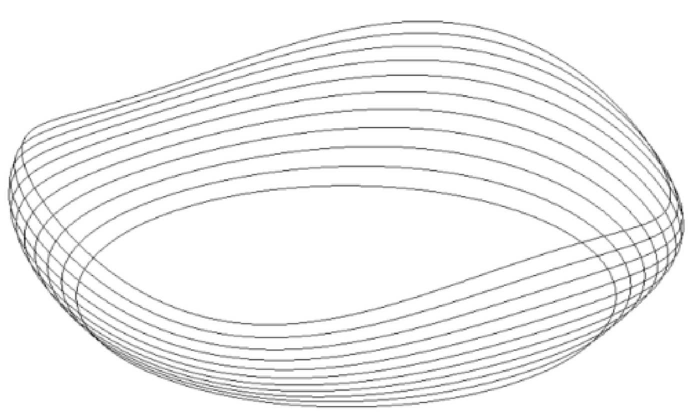
`bowl_pts_h_sin`

Bowl based on an h-dependent sinusoid, which influences each floor's radius.

```
""" Bowl based on an h-dependent sinusoid, which influences each floor's radius. """
bowl_pts_h_sin(p, r, r_amp, h, n, m) =
    map_division((ϕ, h0) -> p + vcyl(r+r_amp*sin(.7π/h*h0), ϕ, h0),
                            # radius' base frequency: .7π in a period of h_max
                                  0, 2π, n,
                                  0, h, m)
```

▶ Generate bowl matrix lines ☐

```
run_test(bowl_pts_h_sin_test) do
    delete_all_shapes()
    map(spline, bowl_pts_h_sin(u0(), 20, 5, 10, 100, 10))
end
```



Geometry

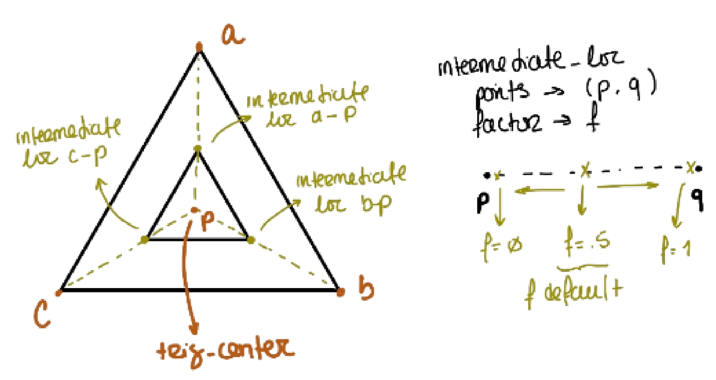
Iterate geometry over point matrices

Façade triangles

`trig_window`

From 3 locs in space, produces a triangular surface with a triangular hole in the middle. Hole size depends on the f factor, which is a value from 0 (no hole) to 1 (no surface).

```
""" From 3 locs in space, produces a triangular surface with a triangular hole in the middle. \\
Hole size depends on the f factor, which is a value from 0 (no hole) to 1 (no surface). """
trig_window(a, b, c, f=.4) =
    let p = trig_center(a, b, c)
        KhepriBase.b_surface_polygon_with_holes(current_backend()[1],
                                                [a, b, c],
                                                [[intermediate_loc(p,a,f),
                                                intermediate_loc(p,b,f),
                                                intermediate_loc(p,c,f)]], -1)
    end
```



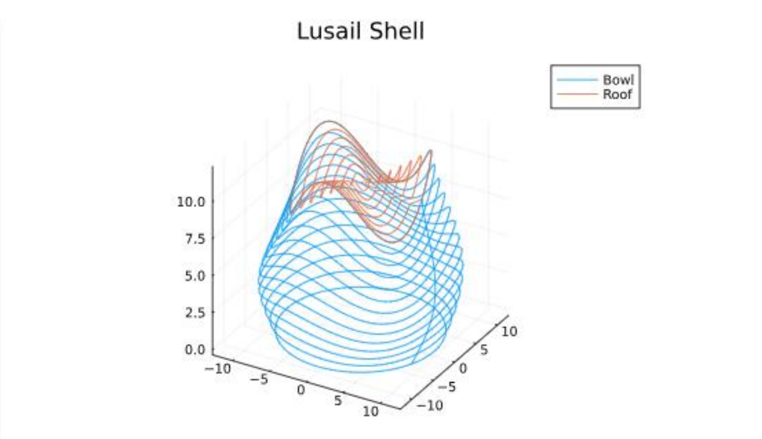
Complete Shell Test

`shell`

Parameters
p = bowl center at the base
r min = roof whole radius
r = bowl radius
r amp = bowl curvature amplitude
h = bowl height
ωh = frequency in height
ωφ = frequency around the bowl (from 0 to 2π)
nb = n points around the bowl (from 0 to 2π)
mb = m points in height on the bowl
nr = nr points around the roof (from 0 to 2π)
mr = mr points along the radius on the roof

Interactive test

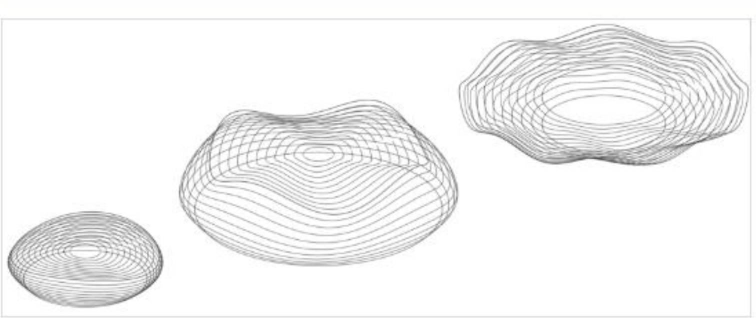
- r = 10
- r min = 0
- r amp = 1
- h = 10
- ωh = 1.0
- ωφ = 1.0
- mb = 10
- mr = 5

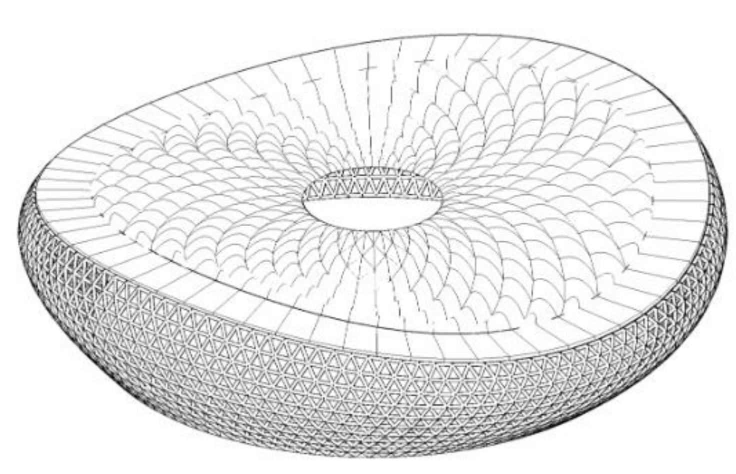



```
let pts_cyl = shell(u0(), r_min_t, rt, r_amp_t, ht, ωht, ωϕt, 100, mbt, 100, mrt)
    pts_bowl = vcat(pts_cyl[1]...)
    pts_roof = vcat(pts_cyl[2]...)
    b_xs, b_ys, b_zs = cx.(pts_bowl), cy.(pts_bowl), cz.(pts_bowl)
    r_xs, r_ys, r_zs = cx.(pts_roof), cy.(pts_roof), cz.(pts_roof)
    plot([b_xs, r_xs], [b_ys, r_ys], [b_zs, r_zs],
        title="Lusail Shell", label=["Bowl" "Roof"])
end
```

▶ Generate shell matrix lines ☐

```
run_test(shell_test) do
    delete_all_shapes()
    map(ptss->spline.(ptss), shell(u0(), 5, 20, 5, 10, 1, 2, 100, 10, 66, 10))
    map(ptss->spline.(ptss), shell(x(90), 5, 40, 5, 20, 2, 4, 100, 10, 100, 10))
    map(ptss->spline.(ptss), shell(x(200), 20, 20, 30, 10, 1/2, 8, 100, 10, 66, 10))
end
```





The Algorithmic Design Sketchbook

Designing buildings programmatically

Algorithmic Design (AD)

AD generates architectural designs through algorithms. With AD, the architect does not create the building's digital model directly, but instead, creates the program that creates the model. It helps:

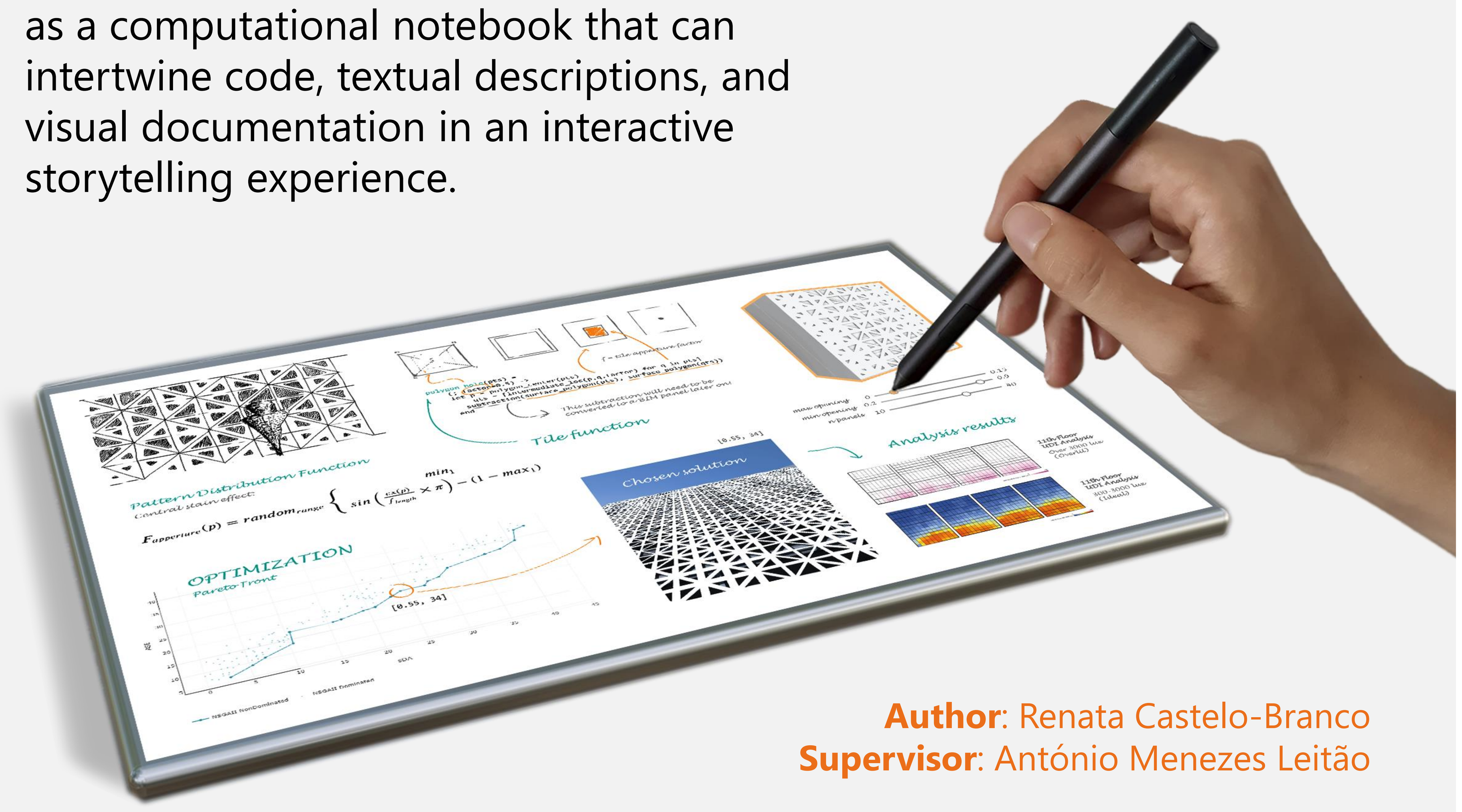
- explore more design solutions
- automate design tasks
- integrate changes in later stages
- optimize designs

Problem

Architectural design is strongly based on visual and spatial reasoning, which is not easy to translate into algorithmic descriptions.

Solution: The AD Sketchbook

A design environment that supports the typical architectural design process, where ideas are mostly represented through sketches and diagrams. It is implemented as a computational notebook that can intertwine code, textual descriptions, and visual documentation in an interactive storytelling experience.



Author: Renata Castelo-Branco
Supervisor: António Menezes Leitão

Textual Documentation

Design briefings
Mathematical formulas
Parameter explanations
Function clarifications

Visual Documentation

Hand-made Sketches
Computer-made Images
Automatic Illustrations

Collaboration

Data availability
Reproducibility
Easy-sharing

Liveliness

Interactive evaluation
Reactivity
Interactive visualizers
Traceability

Related Publications

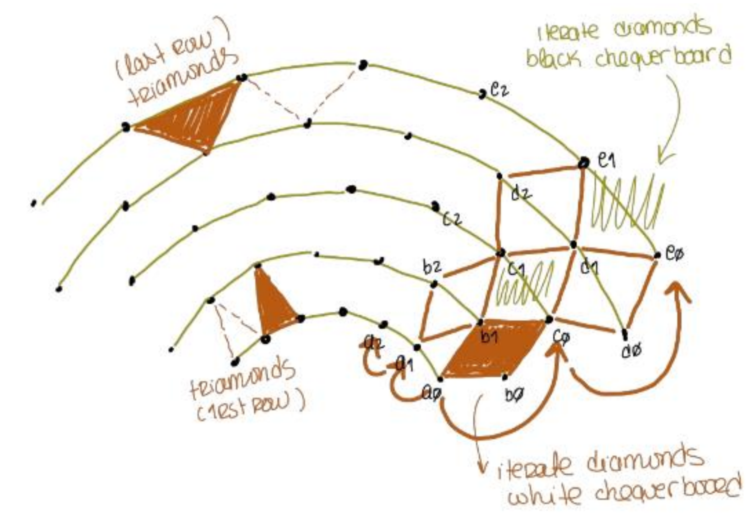
Illustrating Algorithmic Design (2023) CAAD futures
Sketching Algorithmic Design (2022) JAE
Comprehending Algorithmic Design (2022) CAAD futures
Algorithmic Representation Space (2022) Prospectives
Digital Representation Methods: The Case Of Algorithmic Design (2022) FoAR
The Collaborative Algorithmic Design Notebook (2020) ANZAScA
ReAD: Representational Algorithmic Design (2020) ACM <Programming'20>

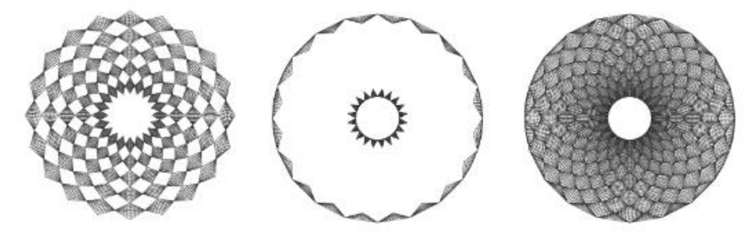
See Also

Lightning Talk given at TNC23: https://www.youtube.com/watch?v=g_EwyEwf_nc
This notebook took 1st place in the **Pluto Notebooks Competition** at JuliaCon2023

Renata's website: https://web.ist.utl.pt/renata.castelo.branco
Our research team: https://algorithmicdesign.github.io/

```
roof_bubbles (generic function with 1 method)
roof_bubbles(p, r_min, r, r_amp, h, ωh, ωϕ, nr, mr) =
    let ptss = roof_pts(p, r_min, r, r_amp, h, ωh, ωϕ, nr, mr)
        ptss_1 = rotate_roof(ptss)
        # first set of bubbles (Black Checkerboard)
        iterate_diamonds(bubble, ptss_1)
        # remove first row to make the second Checkerboard
        ptss_2 = ptss[2:end]
        ptss_2 = rotate_roof(ptss_2)
        # second set of bubbles (White Checkerboard)
        iterate_diamonds(bubble, ptss_2)
        (as, bs) = (ptss_1[1:2])
        (xs, zs) = (ptss_1[end-1:end])
        # first row -> triangular bubbles
        iterate_triamonds(tri_blubble, as, rotate_array(bs, 1))
        # last row -> triangular bubbles
        iterate_triamonds(tri_blubble, zs, rotate_array(xs, 1), -)
    end
```





Superellipse

$$x(t) = a \cdot (\cos^2 t)^{\frac{1}{n}} \cdot sgn(\cos t)$$

$$y(t) = b \cdot (\sin^2 t)^{\frac{1}{n}} \cdot sgn(\sin t)$$

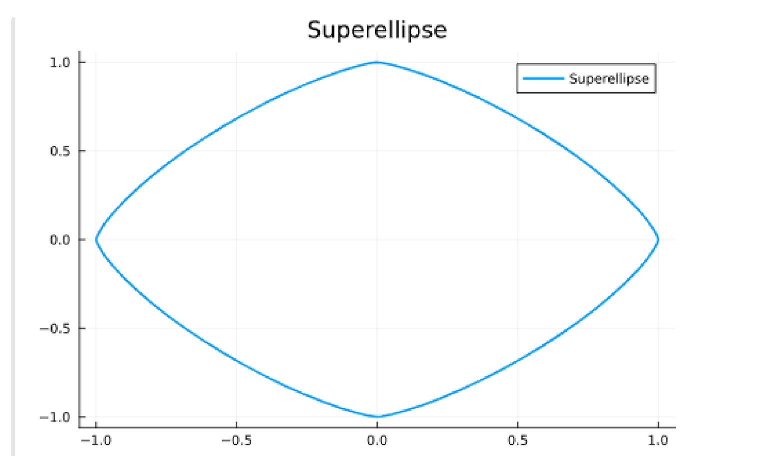
$$-\pi < t < \pi$$

```
superellipse_x (generic function with 1 method)
superellipse_x(a, b, n, t) = a*(cos(t)^2)^(1/n)*sign(cos(t))
```

```
superellipse_y (generic function with 1 method)
superellipse_y(a, b, n, t) = b*(sin(t)^2)^(1/n)*sign(sin(t))
```

Interactive test

- a = 1
- b = 1
- n = 0.1

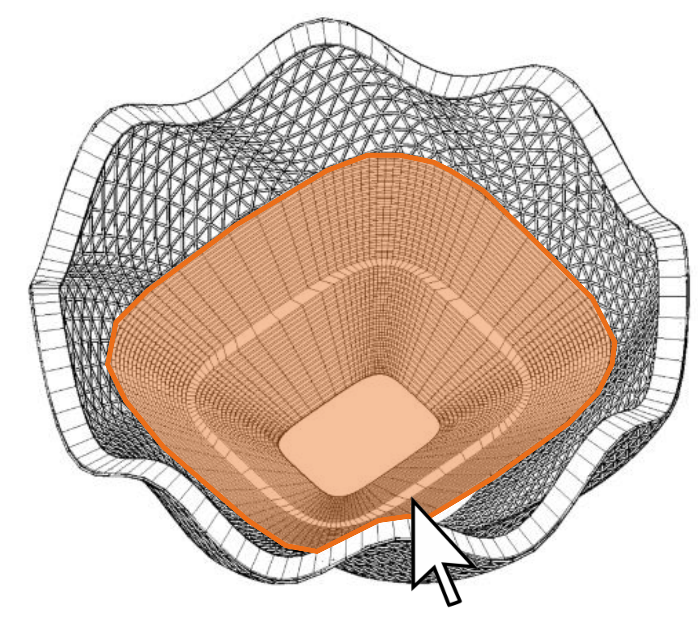



```
let t = range(0, 2π, length=80)
    x = superellipse_x.(a, b, n, t)
    y = superellipse_y.(a, b, n, t)
    plot(x, y, title="Superellipse", label="Superellipse", linewidth=2)
end
```

```
bleachers (generic function with 1 method)
bleachers(p, a, b, n, pts, d, n_str, n_sets) =
    try
        surface_grid(bleachers_mtx(p, a, b, n, pts, d, n_str, n_sets),
            false, true, false)
    catch e
        if isa(e, KhepriBase.BackendError)
            println("Surface grid self-intersection.
            One or more bleach sets were not drawn near the end.")
        end
        println(e)
    end
```

▶ Generate bleachers ☐

```
run_test(bleachers_test) do
    delete_all_shapes()
    bleachers(u0(), 10, 15, 5, 50, .6, 10, 3)
end
```



Complete Stadium

`lusail`

Lusail Stadium

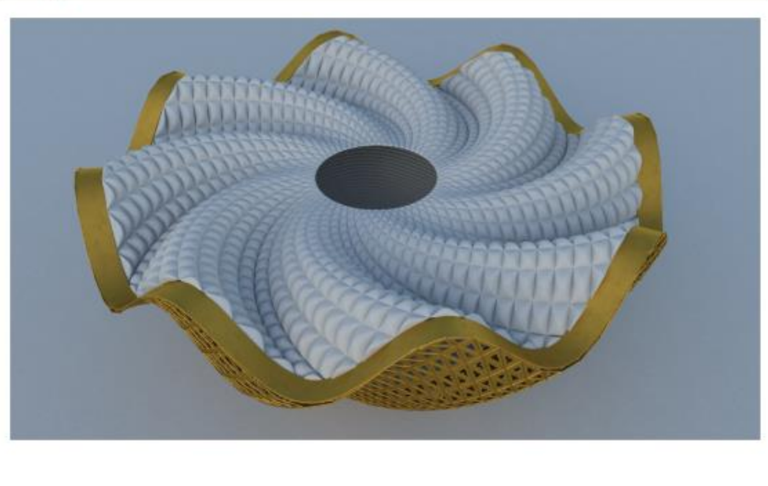
Note: most measures are shrunk for model performance

```
"""
# Lusail Stadium
Note: most measures are shrunk for model performance
"""
lusail(p, r_min, r, r_amp, h, ωh, ωϕ, mb, mr, bd, n_str) =
    let mb = mb < 6 ? 6 : mb # minimum m value (6) for a proper truss
        # make nb depend on mb for (somewhat) equilateral triangles
        nb = round(2π*r*mb/h)
        # make nr depend on mr for (somewhat) balanced diamonds
        nr = round(π*r*mr/r)
        ndiv = find_divisible_num(nb, 50)
        # variables up for change:
        tw, offset, bar_r, surf_t = 1, .2r, .05, .3
        fw, fl, fh, hw, nw, hf = 13, 18, .02, 1, 5, .05
        # end of changeable variables
        r_end = r + sinusoid(r_amp, ωh*.7π/h, 0, h)
        bleacher_l = r_end-fl+1
        n_sets = ceil(Int, min(bleacher_l/bd/n_str, (h-.2h)/bd/n_str))

    with(current_layer, panels) do
        @time facade_panels(p, r, r_amp, h, ωh, ωϕ, nb, mb); end
    with(current_layer, f_bars) do
        @time facade_bars(p, r, r_amp, h, ωh, ωϕ, nb, mb, .1); end
    with(current_layer,truss) do
        @time facade_truss(p, r, r_amp, h, ωh, ωϕ, nb, mb, offset, bar_r, tw, ndiv)
        @time roof_truss(p, r, r_amp, h, ωh, ωϕ, nb, mb, offset, bar_r); end
    with(current_layer, bubbles) do
        @time roof_bubbles(p+vz(h), r_min, r-offset, r_amp, h, ωh, ωϕ, nr, mr); end
    with(current_layer, surf) do
        @time roof_surf(p+vz(h+bar_r/2+surf_t/2),
                        r_min, r, r_amp, h, ωh, ωϕ, nr, mr, offset, surf_t); end
    with(current_layer, concrete) do
        @time bleachers(p-vz(hw), .6fw, .6fl, 5, 50, bd, n_str, n_sets)
        @time base(p-vz(hw), r, hf, 2bar_r, hw, nw, π/10); end
    with(current_layer, grass) do
        @time football_field(p-vz(hw), fw, fl, fh); end
    with(current_layer, ground) do
        @time surface_circle(p-vz(hw-hf), 10r); end

    end
```

Generate variations

▶ Generate variation 3 ☐

```
run_test(var3_test) do
    delete_all_shapes()
    lusail(u0(), 10, 20, 30, 30, 1/2, 8, 20, 20, .5, 20)
end
```



For more on **algorithmic design** and **parametric 3D geometry modeling** visit: https://algorithmicdesign.github.io

inesc id lisboa · TÉCNICO LISBOA · FCT Fundação para a Ciência e a Tecnologia · U LISBOA UNIVERSIDADE DE LISBOA