Provas de conceito em Computação Gráfica

Claudio Esperança
esperanc@cos.ufrj.br

COPPE UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

PESC
Programa de Engenharia de Sistemas e Computação

Uma **prova de conceito**, ou **PoC** (sigla do inglês, *Proof of Concept*) é um termo utilizado para denominar um modelo prático que possa provar o conceito (teórico) estabelecido por uma pesquisa ou artigo técnico. Pode ser considerado também uma implementação, em geral resumida ou incompleta, de um método ou de uma ideia, realizada com o propósito de verificar que o conceito ou teoria em questão é suscetível de ser explorado de uma maneira útil.

https://pt.wikipedia.org/wiki/Prova_de_conceito

... em Computação Gráfica

- Modelos
- Imagens e mídia em geral
- Bibliotecas
- Aplicações
  - Linha de comando / não interativas
  - Interativas

No ensino

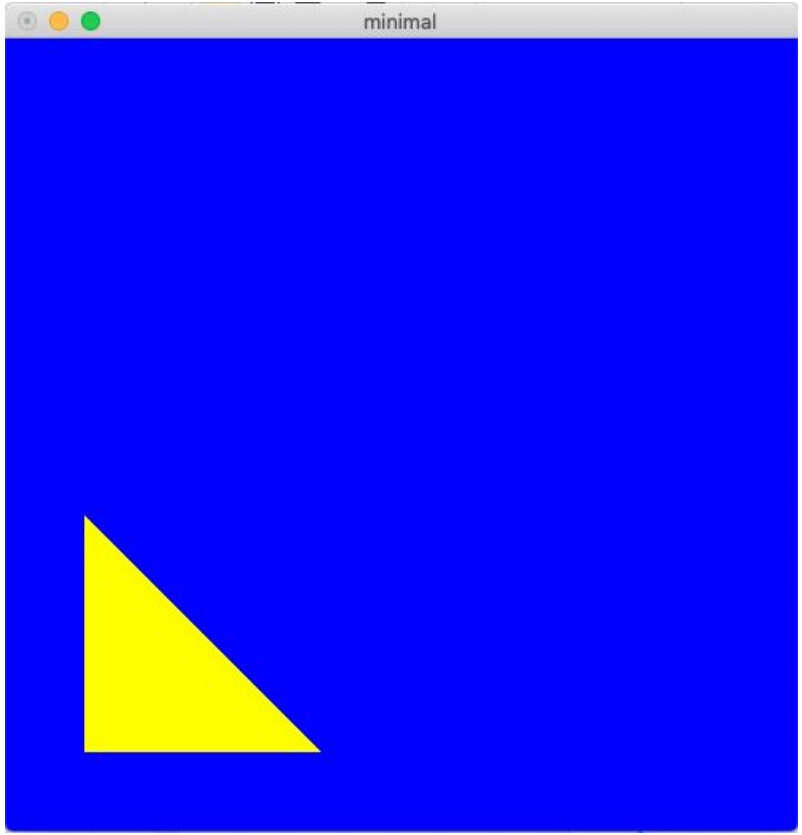
- Motivação
- Ilustração de conceitos
- Exemplos práticos
- Primeiros passos para projetos complexos

Num curso de CG

- Conceitos
  - Transformações e álgebra linear
  - Rasterização
  - Modelagem de curvas e superfícies
  - Animação
- Aplicação
  - Pipeline gráfico
  - Shaders
  - Iluminação
  - Interação

Exemplo: OpenGL hello world

- OpenGL == padrão da indústria
- Como escrever e rodar um “programa mínimo”
- Considerações
  - Que linguagem?
  - Que sistema operacional?
  - Qual versão do OpenGL?



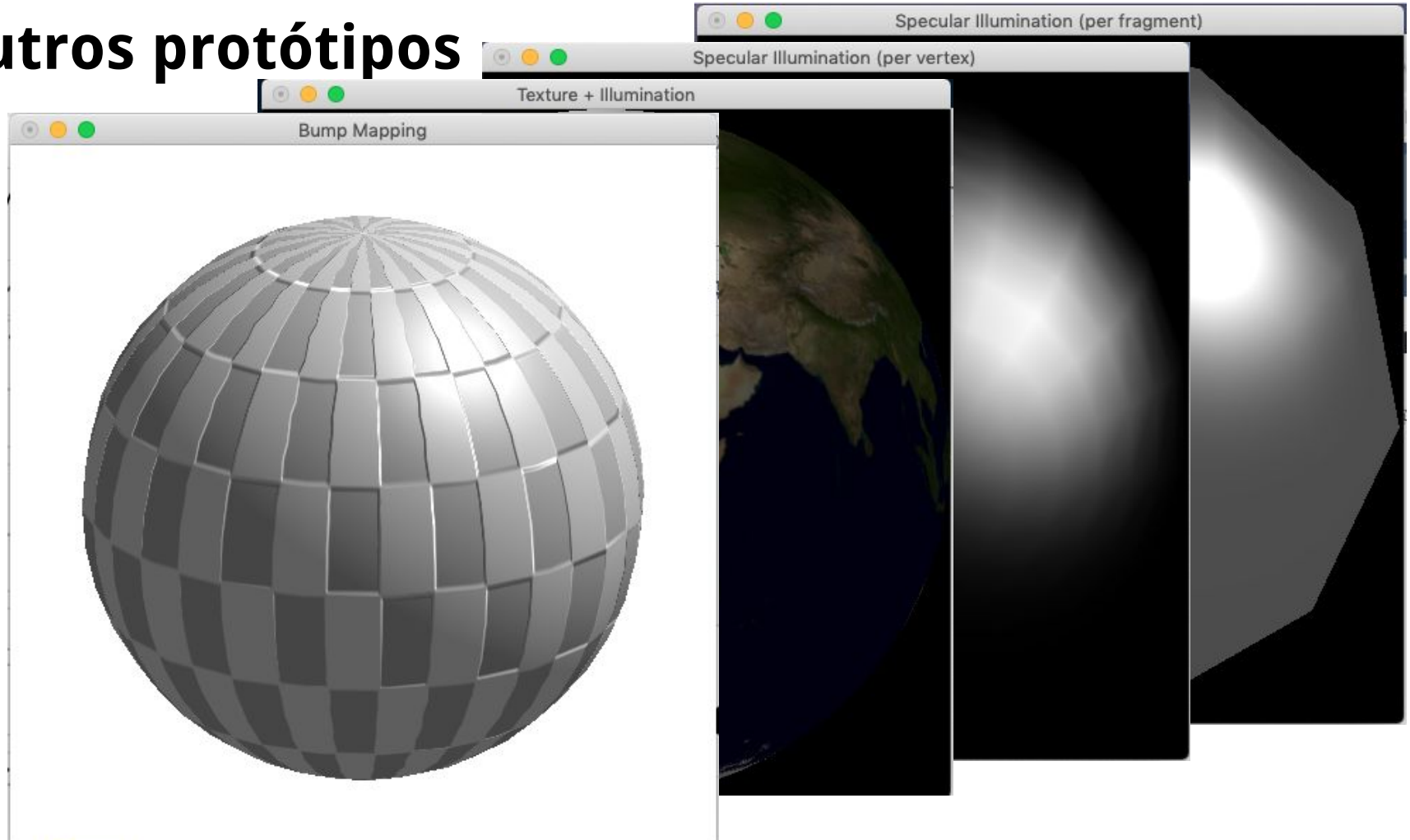
Protótipos OpenGL mínimo

- 5 versões
  - OpenGL 1 / C (GLUT)
  - OpenGL 3 / C++ (shaders)
  - OpenGL 4 / C++ (shaders + GLFW)
  - OpenGL 1 / Python (pyOpenGL)
  - WebGL (~OpenGL ES 2) / JavaScript

Observações

- C/C++
  - Makefiles cuidadosamente montados
  - Bibliotecas GLUT/GLFW/OpenGL instaladas separadamente
- Python
  - Pacote pyOpenGL instalado com gerenciador pip
- WebGL
  - Roda “*as-is*” em navegadores compatíveis

Outros protótipos



Outros protótipos

- Python (shaders)
  - Python é a 1a linguagem aprendida desde 2012 na UFRJ
  - Fácil de distribuir (sem Makefile)
- Interface
  - Mouse usado para alterar parâmetros do modelo de iluminação / textura
- Em sala de aula,
  - Protótipos são modificados em tempo real
  - Alunos são estimulados a contribuir com sugestões

Observações

- Alguns problemas de obsolecência
  - Python 2 deprecado
  - API de biblioteca de imagem alterada
- Interface não intuitiva
  - Uso requer ler texto de instruções
- Numa máquina nova
  - Instalar PIP (package manager)
  - Instalar pyOpenGL
  - Instalar PIL/Pillow (para carregar imagens)

Pesquisa em CG

- Pesquisa reprodutível
  - Ideias científicas publicadas em conjunto com dados e software
- Na prática...
  - Desenvolver protótipos das novas ideias
  - Ter acesso a protótipos das ideias já publicadas
    - Pré-requisitos
    - Comparação

Protótipos de pesquisa em CG

- Geralmente escritos em C++
  - Visam eficiência (tempo/memória)
  - Grande disponibilidade de bibliotecas
  - Linguagem padronizada
- Por vezes,
  - Matlab
    - Muitas ferramentas embutidas
    - Eficiência não crucial
  - Python
    - Grande biblioteca
    - Flexibilidade
    - Fácil de programar
  - Outras

Exemplos de protótipos

Contents lists available at ScienceDirect

Computers & Graphics

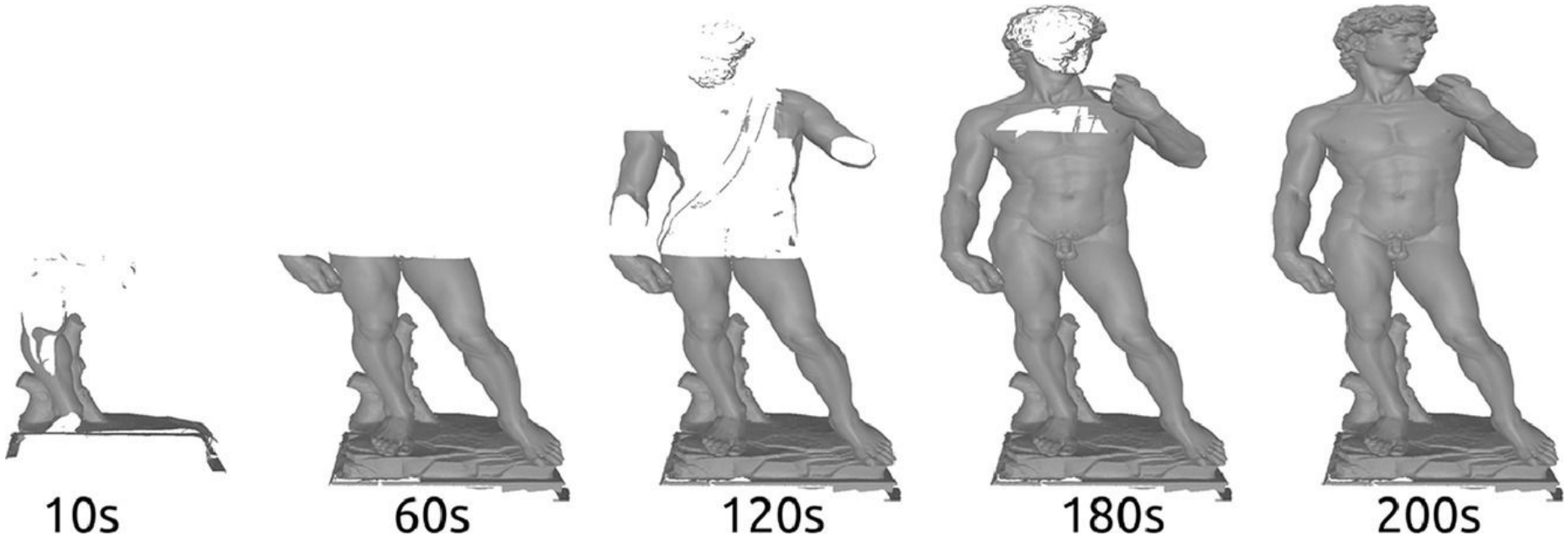
journal homepage: www.elsevier.com/locate/cag

OMiCroN - Oblique Multipass Hierarchy Creation while Navigating

Vinícius da Silva[a,*], Claudio Esperança[b], Ricardo Marroquim[b,c]

[a] *Institute for Pure and Applied Mathematics (IMPA). VISGRAF Lab. Estrada Dona Castorina, 110, Jardim Botânico, Rio de Janeiro - RJ, Brazil, CEP: 22460-320*
[b] *Federal University of Rio de Janeiro (UFRJ). Computer Graphics Lab (LCG). Cidade Universitária, Centro de Tecnologia, Block H, Rio de Janeiro - RJ, Brazil, CEP: 21941-972*
[c] *Delft University of Technology (TU Delft). Computer Graphics and Visualization Group. Van Mourik Broekmanweg 6, Delft, The Netherlands*

doi:10.1016/j.cag.2019.08.016



Estratégias para construir hierarquias com renderização “on the fly”

Protótipo OMiCRoN

- Código disponível em https://github.com/dsilvavinicius/OMiCroN
- C++ e OpenGL
- STXXL (stxxl.org - STL for extended datasets)
- Tucano (https://gitlab.com/lcg/tucano - OpenGL framework)
- Comparação com outras estratégias

[42] Schütz, M. Potree: Rendering large point clouds in web browsers. Technische Universität Wien, Wiedeń 2016;.

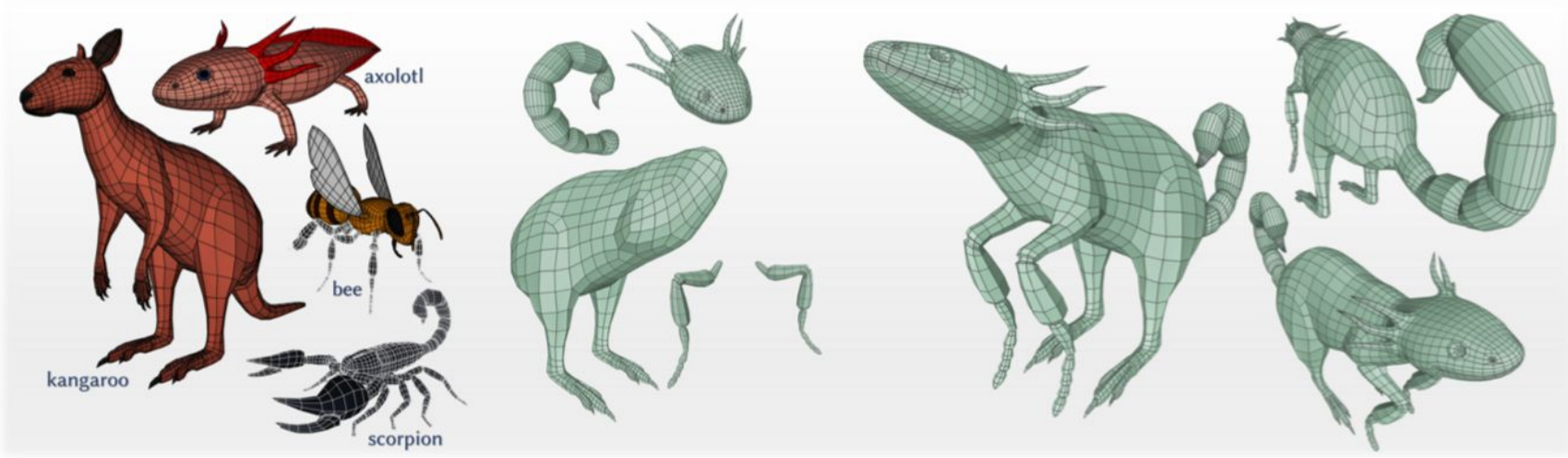
[43] Baert, J, Lagae, A, Dutr, P. Out-of-core construction of sparse voxel octrees. Computer Graphics Forum 2014;33(6):220–227. URL: http://dx.doi.org/10.1111/cgf.12345. doi:10.1111/cgf.12345.

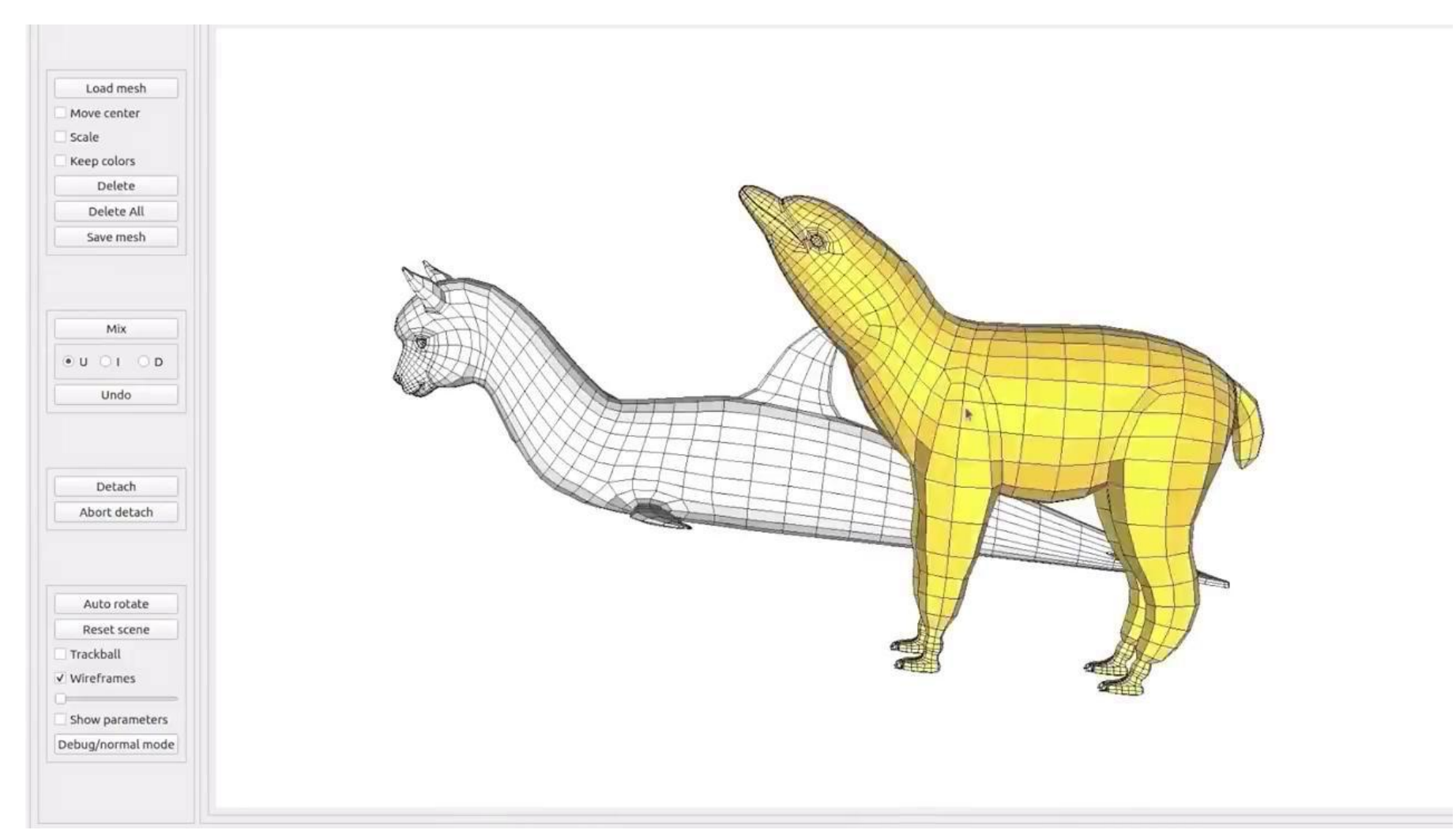
...poucas implementações em condições de reuso

QuadMixer: Layout Preserving Blending of Quadrilateral Meshes

STEFANO NUVOLI, University of Cagliari, Italy
ALEX HERNANDEZ, Federal University of Rio de Janeiro, Brazil
CLAUDIO ESPERANÇA, Federal University of Rio de Janeiro, Brazil
RICCARDO SCATENI, University of Cagliari, Italy
PAOLO CIGNONI, CNR of Italy, Italy
NICO PIETRONI, University of Technology Sydney, Australia

Nuvoli, Stefano; Hernandez, Alex; Claudio Esperança; Scateni, Riccardo; Cignoni, Paolo; Pietroni, Nico. QuadMixer: Layout preserving blending of quadrilateral meshes. doi:10.1145/3355089.3356542. *Acm Transactions On Graphics*. 38 pp.1-13, 2019.



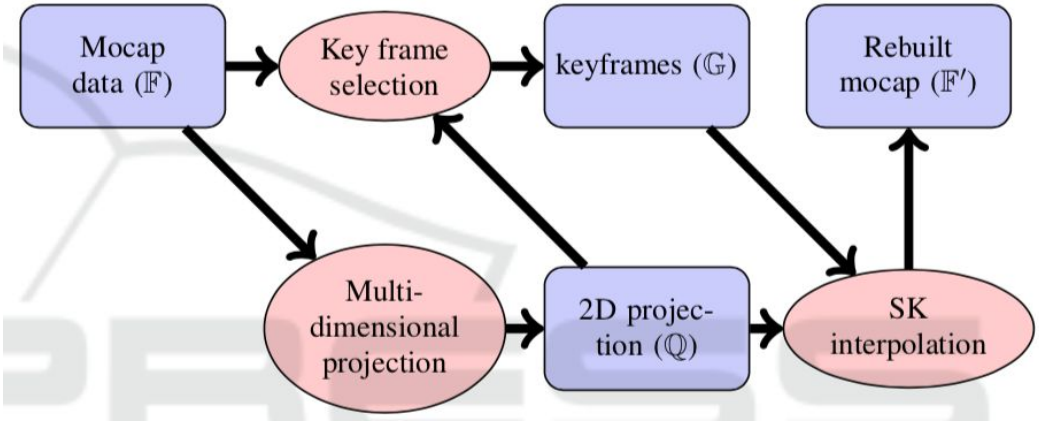

Load mesh
Move center
Scale
Keep colors
Delete
Delete All
Save mesh
Mix
U
I
D
Undo
Detach
Abort detach
Auto rotate
Reset scene
Trackball
Wireframes
Show parameters
Debug/normal mode

Protótipo Quadmixer

- C++
- Gurobi
- VCG
- CG3Lib
- libIGL
- CGAL
- Qt
- Código Quadrangulação [Takayama] parcialmente reimplementado
- “Make” bastante difícil

> We performed our test on a desktop computer with an Intel i7-8750H processor with 16GB of RAM. We used Gurobi [Gurobi Optimization 2018] to solve the minimization of Section 3.3. All the code is single-threaded and not highly optimized; it has been implemented using the VCG Library [CNR 2013], CG3Lib [Muntoni et al. 2019], libigl [Jacobson et al. 2013b], and CGAL [The CGAL Project 2019].

Enhancing Spatial Keyframe Animations with Motion Capture



Costa, Bernardo; Claudio Esperança. Enhancing Spatial Keyframe Animations with Motion Capture. *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications.* doi:10.5220/0007296800310040. pp. 31-40, 2019.

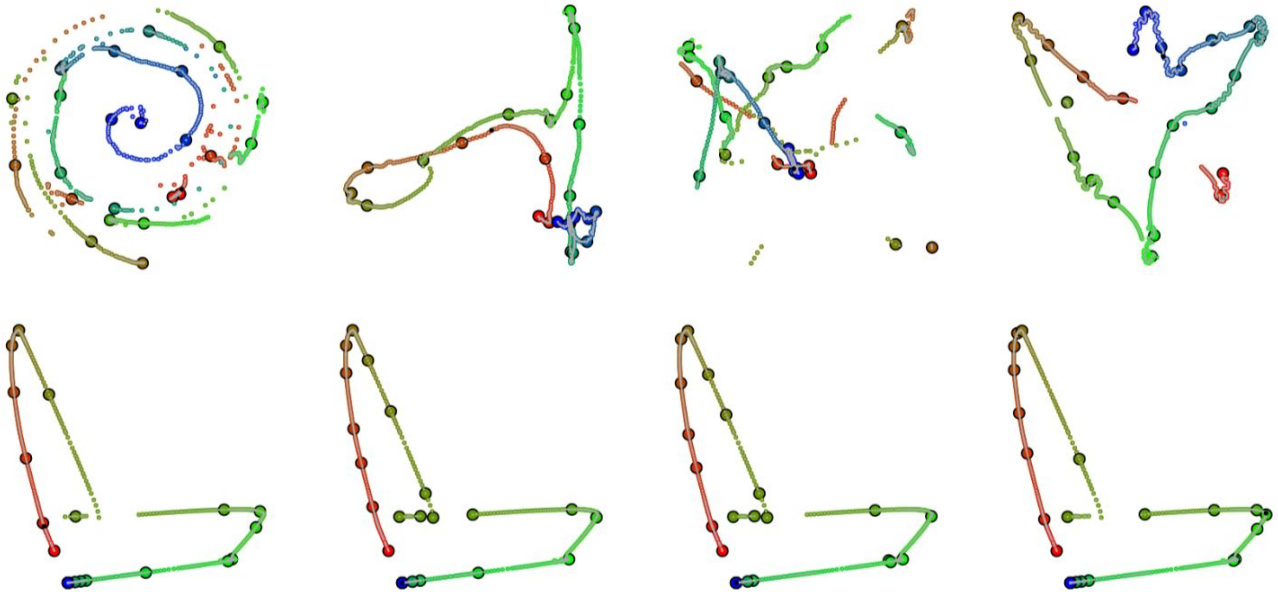


Figure 1: Implemented projection and keyframe extraction algorithms. Results obtained by projecting the poses in file `05_10.bvh` from the CMU Motion Capture database. The color gradient from red to green to blue is used to indicate time. Bigger dots with black border are keyframe poses (markers). First row from left to right: Force, MDS, PCA and t-SNE, where 3% of the poses are selected as keyframes by uniform sampling (US). Second row shows the results for LLE with keyframes selected with US, SCS, PB and ROLS.

Protótipo SKF-Mocap

- Algoritmos escritos em C++
  - Biblioteca Eigen
  - Reconstrução com RBFs
  - 6 algoritmos de projeção multidimensional + variantes
  - Código disponível em https://gitlab.com/bfcosta/projmocapskf
- Visualização de trajetórias escrito em JS
- Grande quantidade de experimentos
  - Artigo ⇒ resultados sumarizados
  - Métricas de qualidade de reconstrução não condizentes com resultados visuais

Considerações gerais

- Disponibilizar código é mais crítico do que disponibilizar mídia
- Código publicado não necessariamente é código aproveitável
- Dependências são muitas vezes incontroláveis
- Protótipos interativos são recomendáveis mas nem sempre exequíveis
- Interatividade adiciona uma camada considerável de complexidade
- Idealmente, código executável (sem make)

Visualização de dados

- Área de crescente interesse na comunidade de CG

- Pesquisa gera dados. Como apresentá-los?
  - Tabelas
  - Gráficos
  - Vídeos
  - Aplicações
    - Off-line
    - Online

Ferramentas para Visualização de Dados

- Gráficos offline (dados ⇒ imagem)
  - Office suites
  - Gnuplot
  - Matplotlib (Python)

- Gráficos online
  - Plataformas (dados ⇒ visualização)
    - Tableau
    - Flourish
    - Muitos outros
  - Web apps (aplicações)

Web apps vs Desktop apps

Acesso padronizado ao HW	Depende apenas de drivers

- Browser as SO
- Câmera, microfone, áudio, framebuffer, webmidi
- Sem acesso direto ao sistema de arquivos
- Limitações de segurança

Web apps vs Desktop apps

Acesso padronizado ao HW	Depende apenas de drivers
Memória gerenciada pelo navegador	Memória gerenciada pelo SO

- Garbage collection
- Caching

Web apps vs Desktop apps

Acesso padronizado ao HW	Depende apenas de drivers
Memória gerenciada pelo navegador	Memória gerenciada pelo SO
Pode ser executado localmente ou a partir de um link remoto	Executável tem que ser construído (make) ou instalado localmente

Web apps vs Desktop apps

Acesso padronizado ao HW	Depende apenas de drivers
Memória gerenciada pelo navegador	Memória gerenciada pelo SO
Pode ser executado localmente ou a partir de um link remoto	Executável tem que ser construído (make) ou instalado localmente
WebGL	OpenGL

WebGL vs OpenGL

- WebGL 2 ⇔ OpenGL ES 3
  - Alivia restrições do WebGL 1
    - Texturas 3d
    - Texturas não potência de 2
    - Acesso a texels por índices (GPGPU)
- Apenas Vertex e Fragment Shaders
- Única maneira padronizada para acesso à GPU
  - Mas vejam WebGPU (https://gpuweb.github.io/gpuweb/)

Web apps vs Desktop apps

Acesso padronizado ao HW	Depende apenas de drivers
Memória gerenciada pelo navegador	Memória gerenciada pelo SO
Pode ser executado localmente ou a partir de um link remoto	Executável tem que ser construído (make) ou instalado localmente
WebGL	OpenGL
JavaScript / WebAssembly	Qualquer linguagem

JavaScript

- Linguagem de programação nativa de navegadores modernos
- Na verdade, ECMAscript

- Originalmente, usada para manipular o DOM
- Modernamente, linguagem de propósito geral
  - Node.js

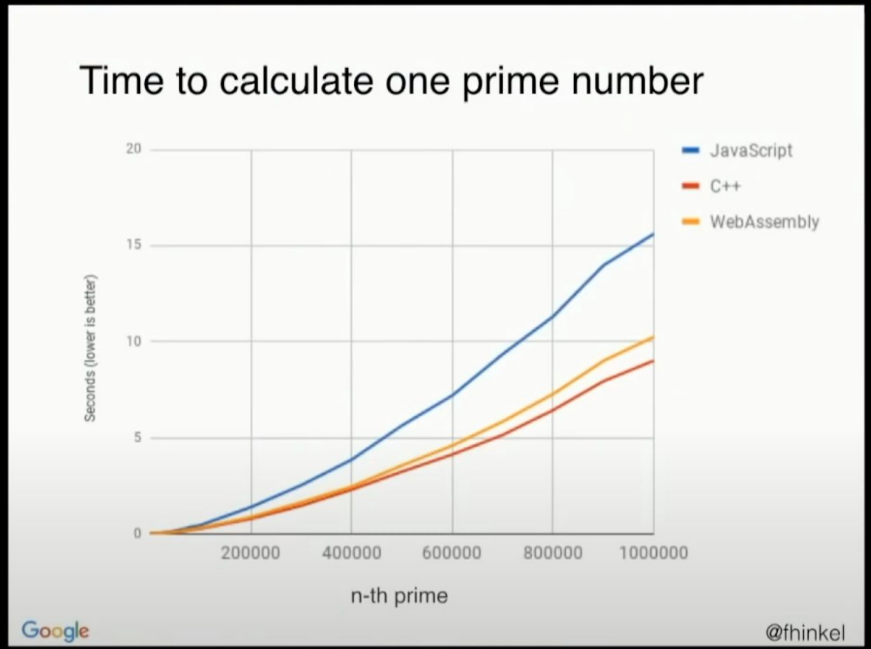
- Linguagem script, funcional, single-threaded, assíncrona
- Implementada com compilador JIT (Just In Time)

WebAssembly

WebAssembly is a new type of code that can be run in modern web browsers — it is a low-level assembly-like language with a compact binary format that runs with near-native performance and provides languages such as C/C++ and Rust with a compilation target so that they can run on the web. It is also designed to run alongside JavaScript, allowing both to work together.

https://developer.mozilla.org/en-US/docs/WebAssembly

JavaScript vs C++ vs WebAssembly: Speed, Speed, Speed



27:05 / 28:49

Scroll for details

https://youtu.be/uMuYaES4W3o

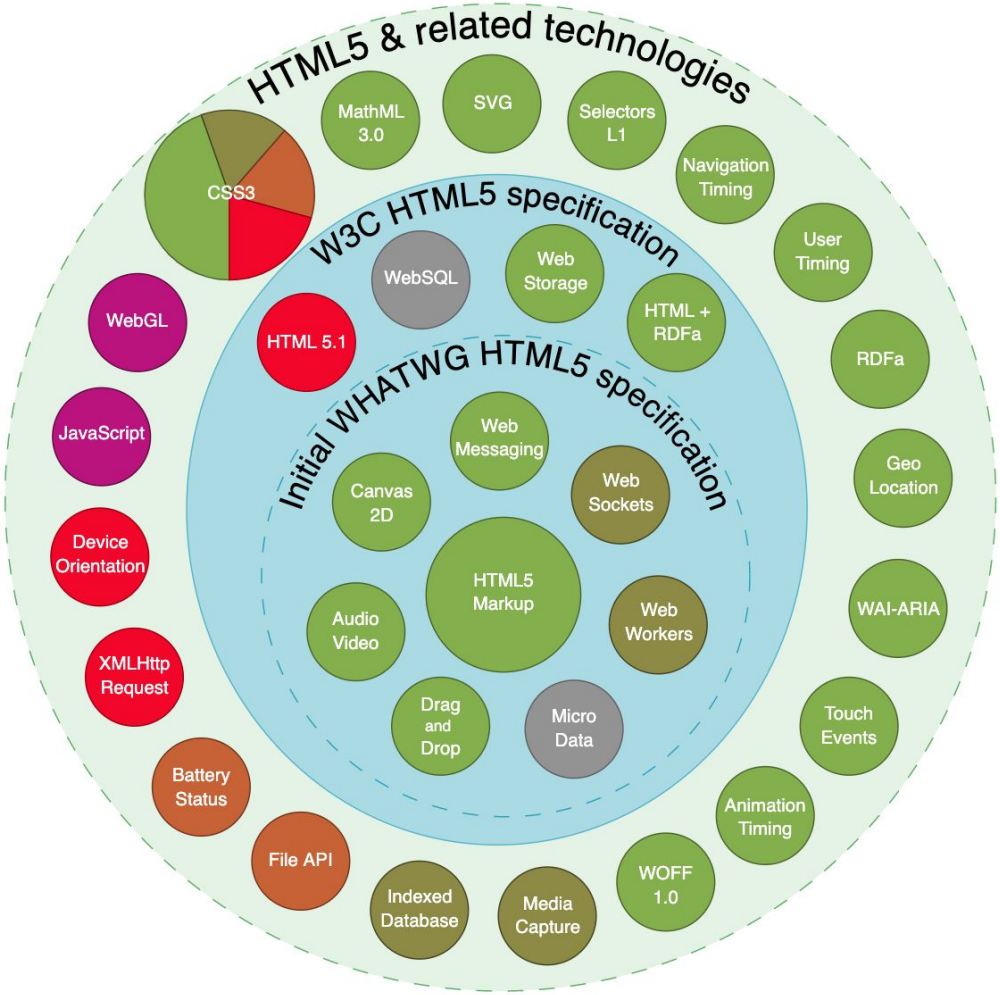
Back-end e Front-end

- Back-end
  - Código que roda no servidor web
  - Minimamente apenas um servidor web que disponibiliza arquivos
  - Opcionalmente gera conteúdo “on-the-fly”
    - PHP, C, Python, Java, JavaScript
    - Bancos de dados
- Front-end
  - Conjunto de tecnologias conhecido como HTML5
  - Linguagem de programação: JavaScript

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



HTML5 e Computação Gráfica

- SVG (Scalable Vector Graphics)
  - Gráficos vetoriais
  - Markup (grafo de cena)
- Canvas
  - Gráficos matriciais (modo imediato)
  - APIs
    - Canvas 2D
    - WebGL (OpenGL ES)

Desenvolvendo web apps

- O navegador já tem um console JS (*inspect!*)
- Código JS embutido em documentos html usando tag <script>

```
<script>
   document.write("Hello World!")
</script>
```

- Grande número de APIs
- Grande número de bibliotecas
- Grande número de editores/IDEs offline e online

Observable

observablehq.com

Plataforma de desenvolvimento iniciada por Mike Bostock (d3)

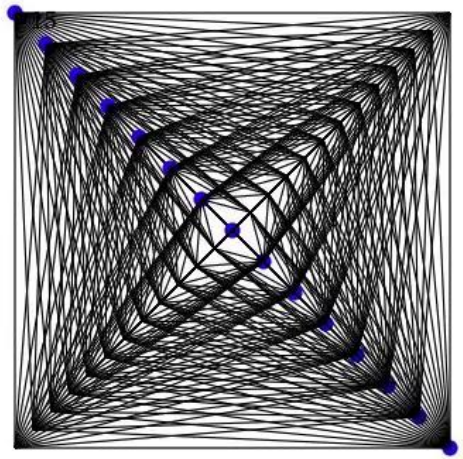
- Apps ⇔ notebooks
  - Literate programming
  - Foco pedagógico
- Programação reativa
  - Código encapsulado em células
  - Grafo de dependências
  - Reavaliação automática

Live coding

Um notebook de teste

```
pointsOnLine = f(p, q, n)
```

```
pointsOnLine = function(p,q,n) {
  let r = []
  for (let [x1,y1] = p, [x2,y2] = q, i = 0; i < n; i++) {
    r.push([x1+(x2-x1)/(n-1)*i,y1+(y2-y1)/(n-1)*i])
  }
  return r
}
```



15

25

https://observablehq.com/d/2a752020f05cfd76

Demos de conceitos

Published Apr 12

Transformation demo

An interactive demo for experimenting with 2D transformation matrix composition.

+ Translate | + Scale | + Rotate | + Shear | Reset

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

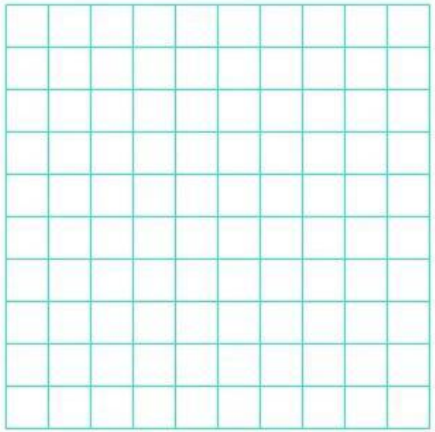
https://observablehq.com/@esperanc/transformation-demo

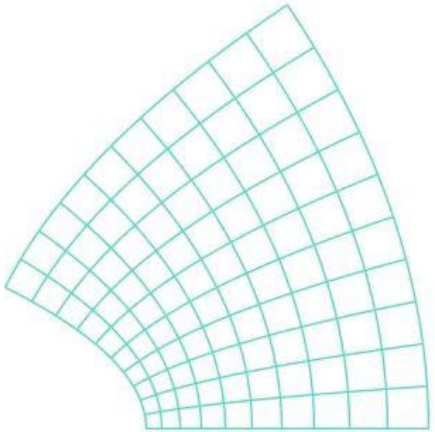
observablehq.com/@renatoppl/conformal-maps

```
ConformalMona2(d => Complex.exp(d), 0)
```

Other interesting functions below (you can draw on the left in all cases):

$$f(z) = z + z^2$$

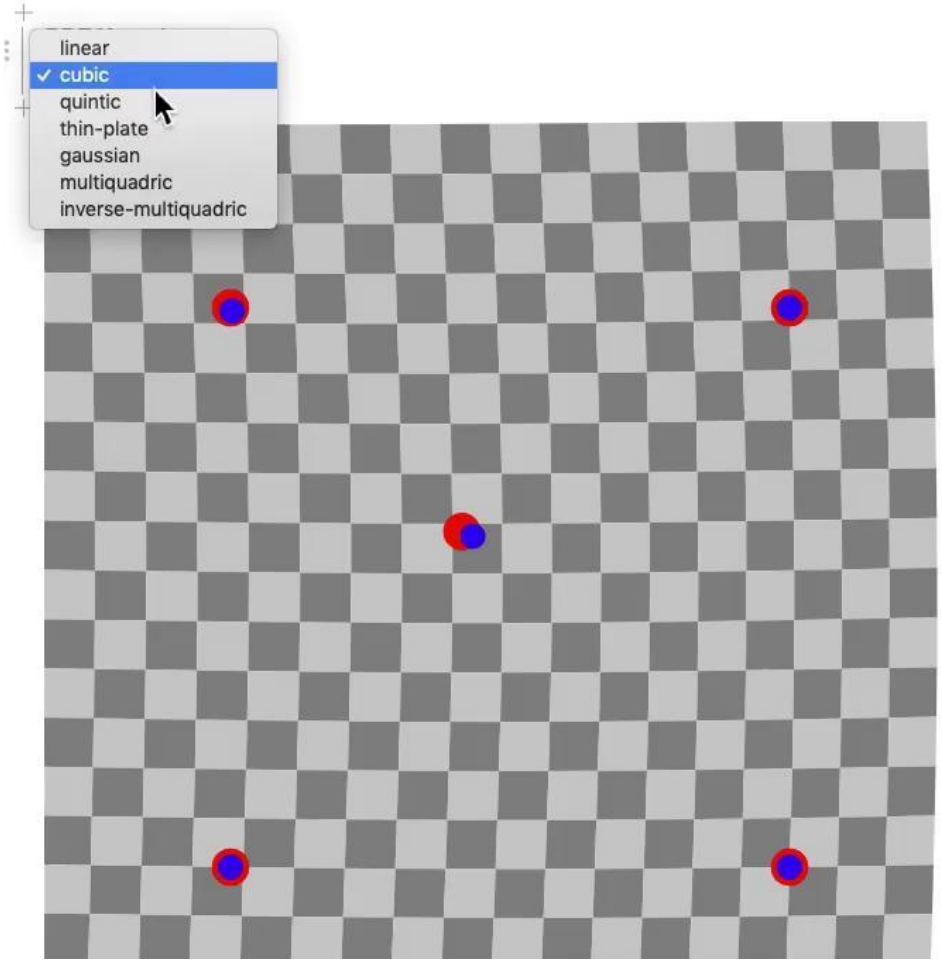




```
ConformalTransformation(p => Complex.sum(p, Complex.prod(p,p)), 0, 450, 270, 80);
```

https://observablehq.com/@renatoppl/conformal-maps

observablehq.com/@esperanc/deformation-with-rbfs



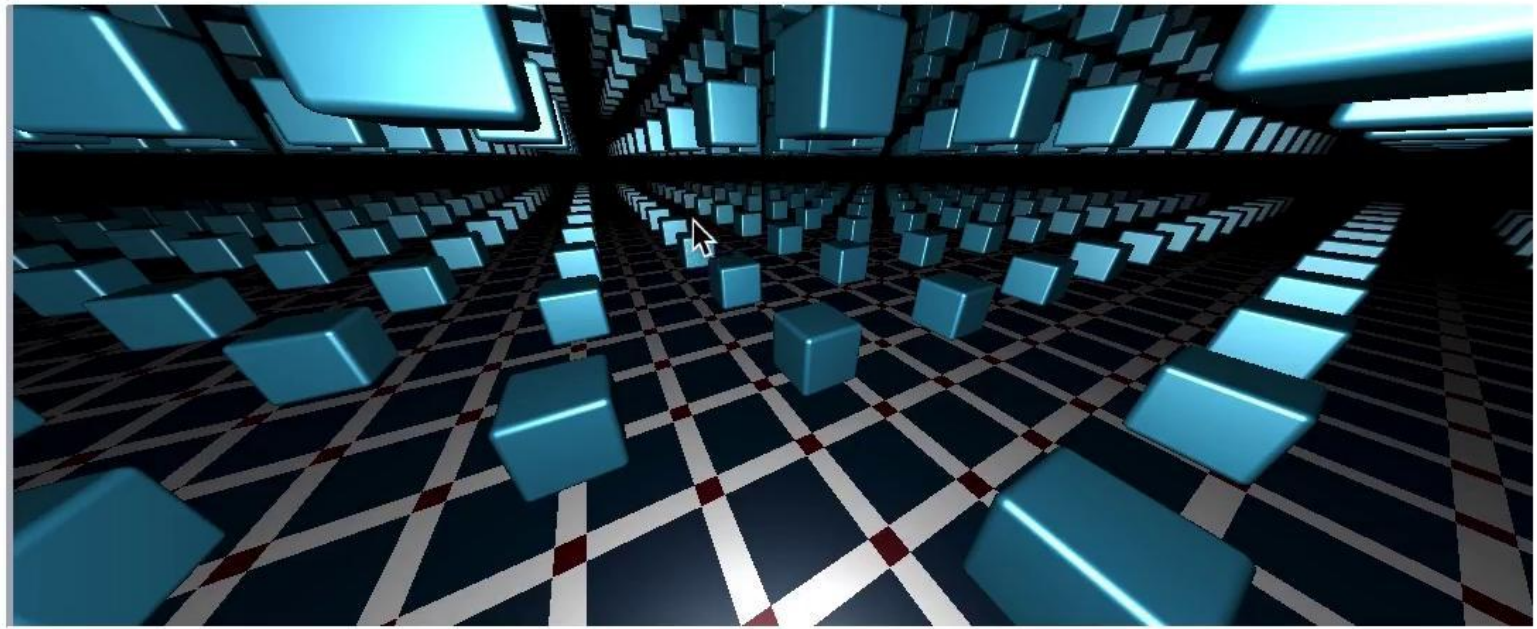
Técnicas

Not Secure | observablehq.com/@makio135/hello-raymarching

GLSL / 10 of 12 ‹ Prev Next ›

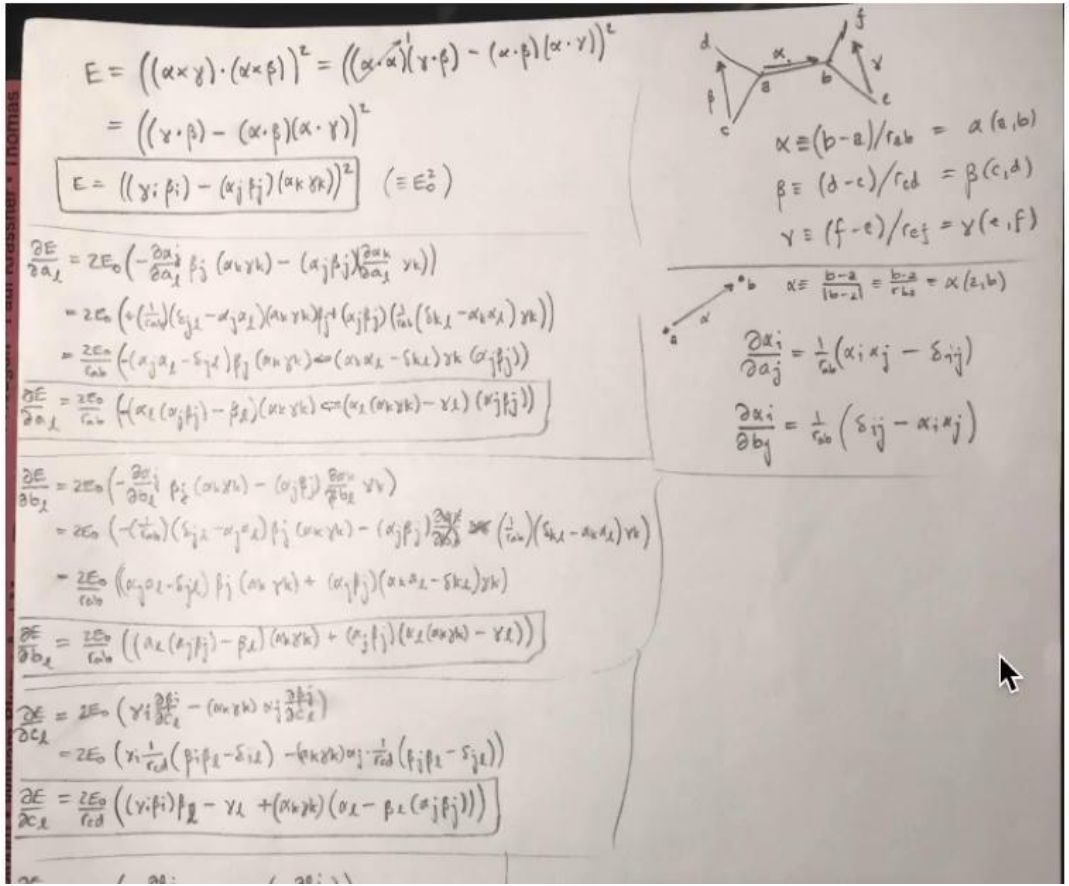
Published Oct 7, 2019 1 Fork Listed in GLSL

Hello Raymarching



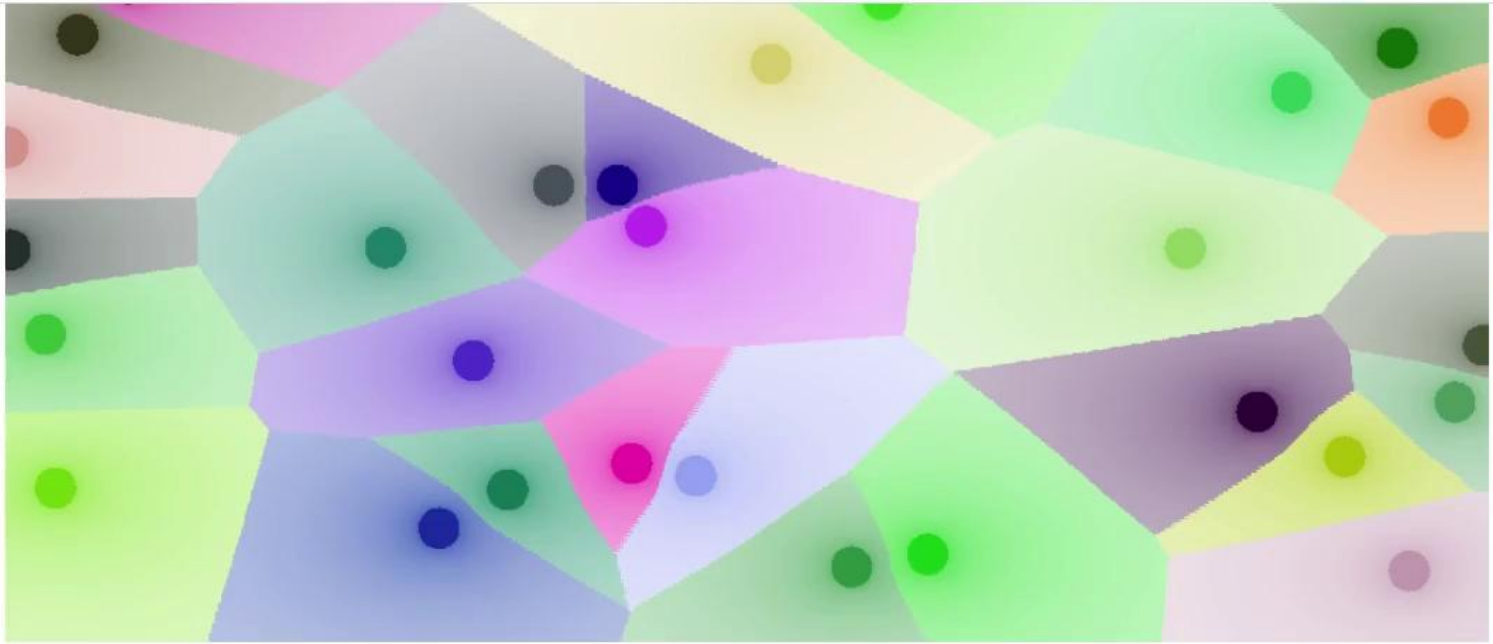
Click to play/pause

https://observablehq.com/@makio135/hello-raymarching



Implementações de artigos

observablehq.com/@esperanc/jump-flooding



Clear | Compute distance field | Demo

Display distance field as

◉ Color gradient ○ Voronoi and medial axis lines

Distance field color attenuation

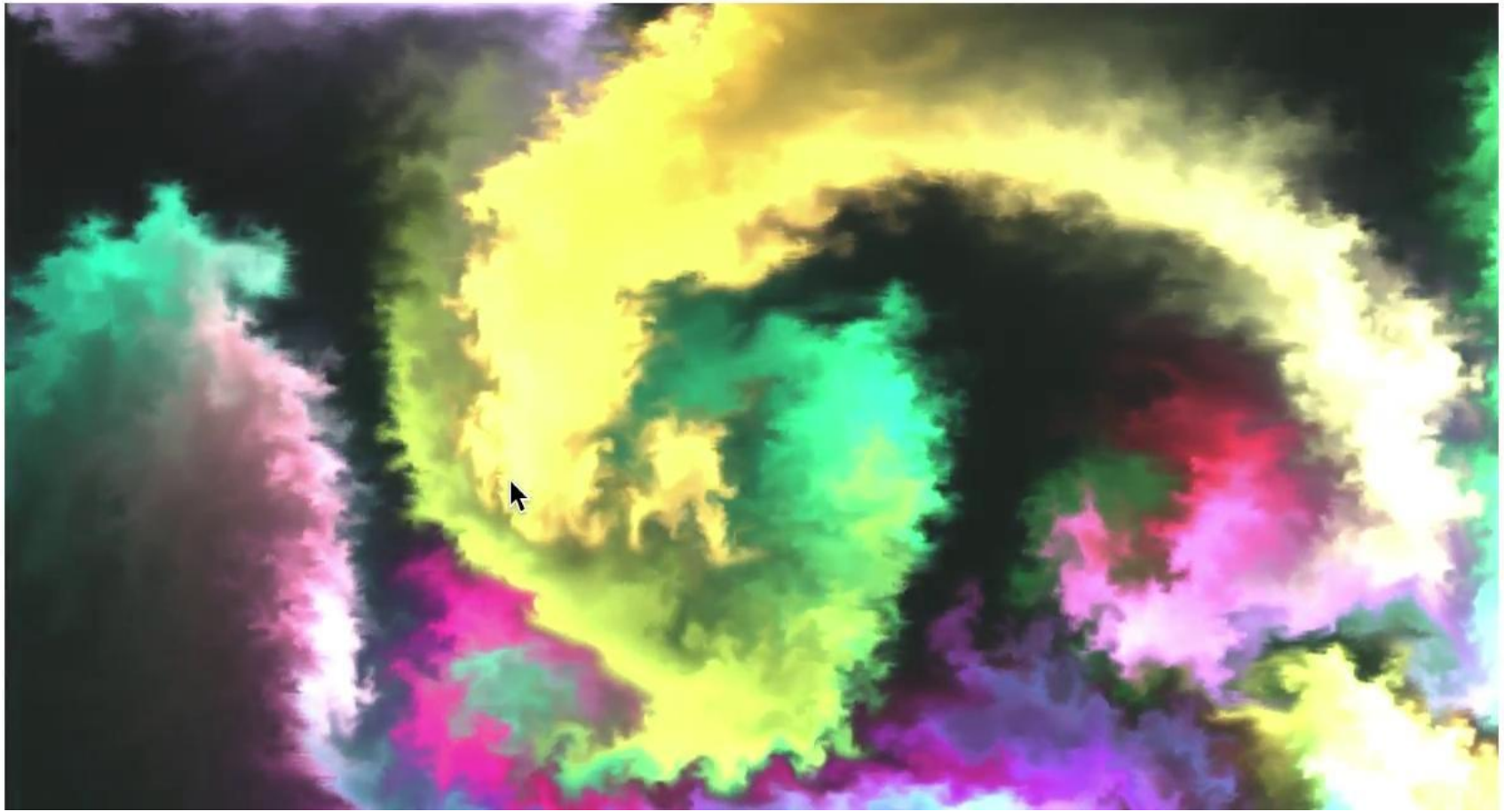
0.829

Controls how much color is attenuated by distance

https://observablehq.com/@esperanc/jump-flooding

observablehq.com/@jashkenas/webgl-fluid-simulation

Click to re-splat the fluid. Click multiple times for *even more fun.*



https://observablehq.com/@jashkenas/webgl-fluid-simulation

observablehq.com/@esperanc/synthesizing-periodic-tilings-of-the-plane

Main interface

Database: Galebach Sá e Sá *Pattern:* t1006

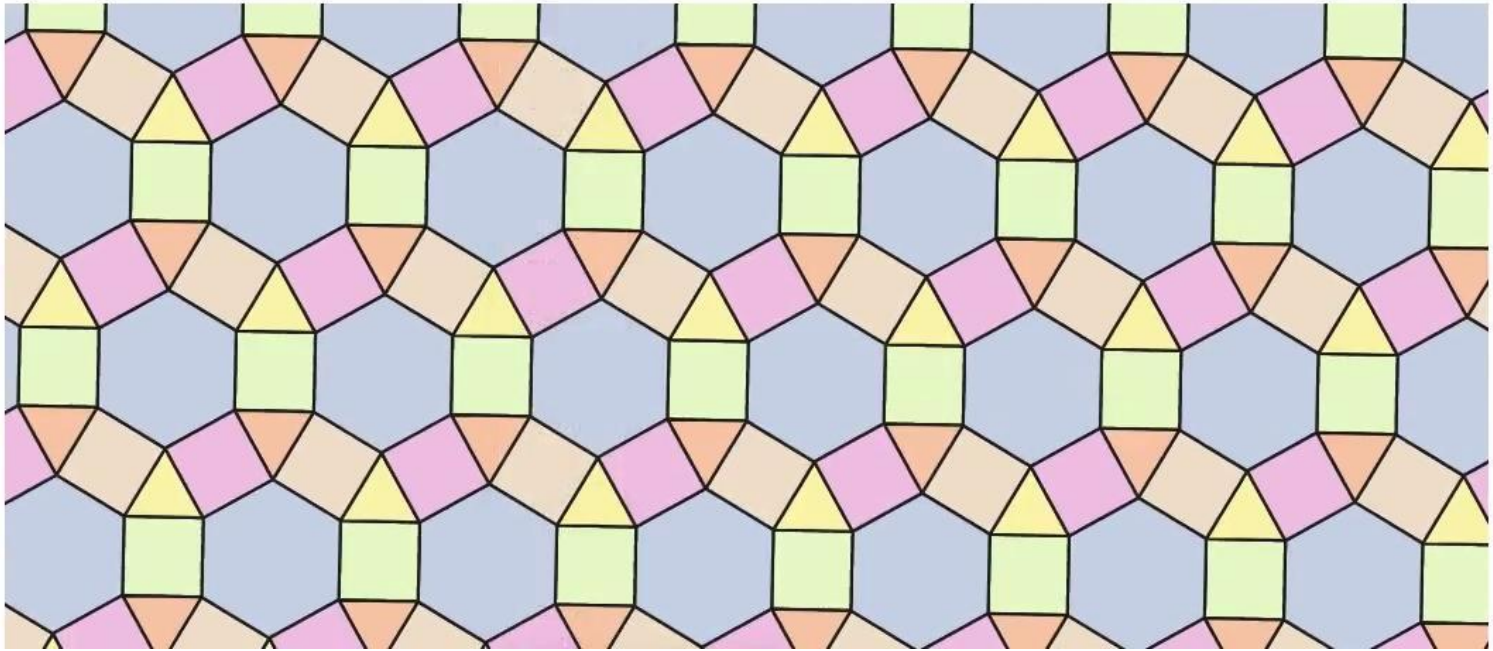
Edge length: 51

Edge thickness: 2

Edge color:

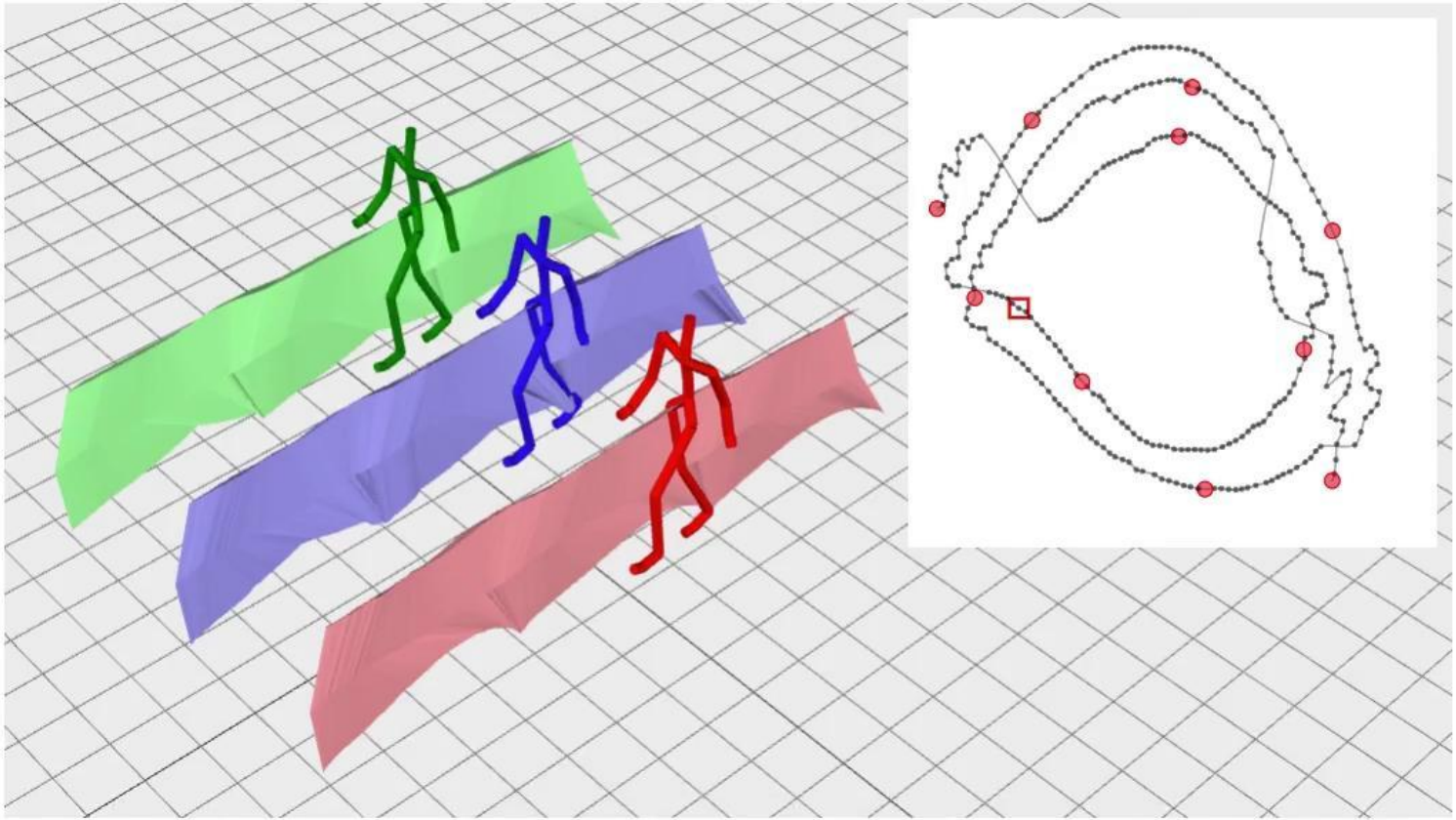
Palette: Pastel2 shift colors

auto-shift auto-rotate



https://observablehq.com/@esperanc/synthesizing-periodic-tilings-of-the-plane

observablehq.com/@esperanc/mocap-projection-browser



Current frame: 125 Animate? ☐

https://observablehq.com/@esperanc/mocap-projection-browser

Últimas observações

Web apps não servem para tudo, mas 1a opção em visualização

Observable é um ambiente bastante produtivo e divertido

HTML5 é não trivial

JS vem se tornando mais fácil de programar

SVG é excelente para gráficos 2d, mas lento e consome memória

Canvas é fácil e rápido para gráficos 2d (processamento de imagem)

WebGL é mais rápido, mas menos flexível para 2d

Obrigado!