Towards Hyperspace

multi-dimensional scene-graphs for data visualization

Lewey Geselowitz, Oct 19th, 2020

Special thanks to...

Plot

lewdo_kernel

world peace*

mostly hyperspace

challenge

proof

*the protocol by which we share space with each other

Proof

Due to Turing-machine equivalency...

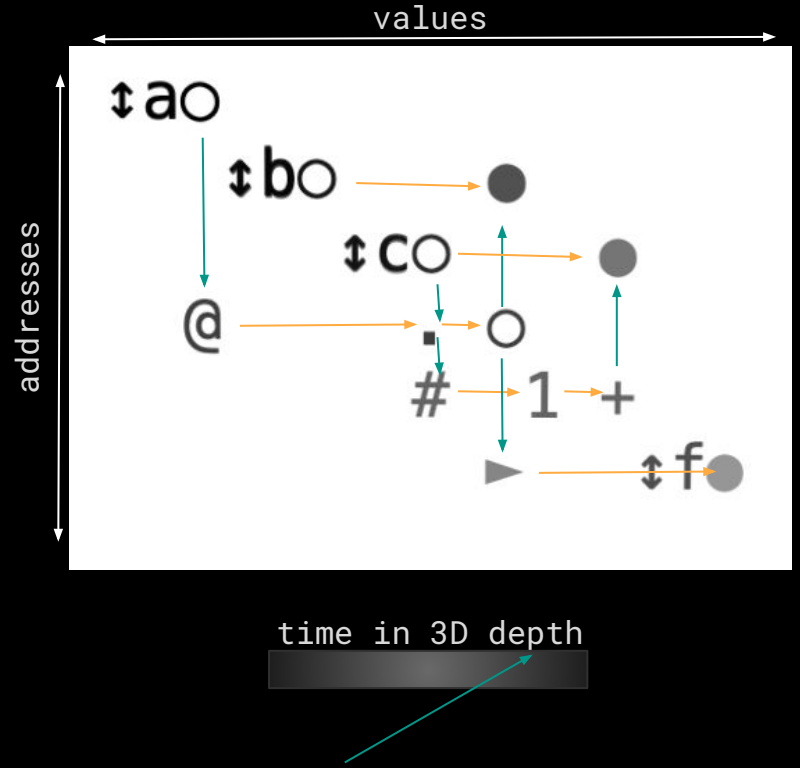
Computation is 3 dimensional:
ADDRESS (=) VALUE (@) TIME

(see *“ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGS PROBLEM”* by Alan Turing, Nov 12 1936)
(or “The Imitation Game”, 2014)

So we could show people everything their computers have done
…but it requires a 3D interface
…and is a big graph ((2^64)^3)

Turing-Machine Visualization

demo link
(best in 3D)



legend

```
3Dtime
↕addresses ↔values

○read ●write .dot
#number @object
+add ►call ~free
```

JavaScript

```
b=a[c];
c=c+1;
var f=b();
```

Challenges

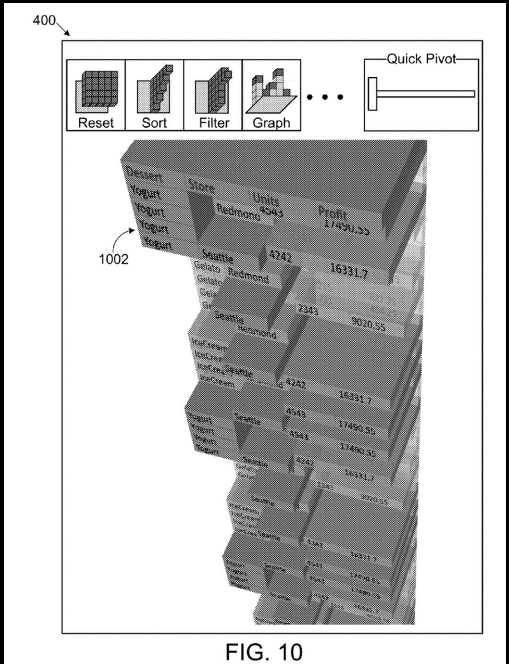
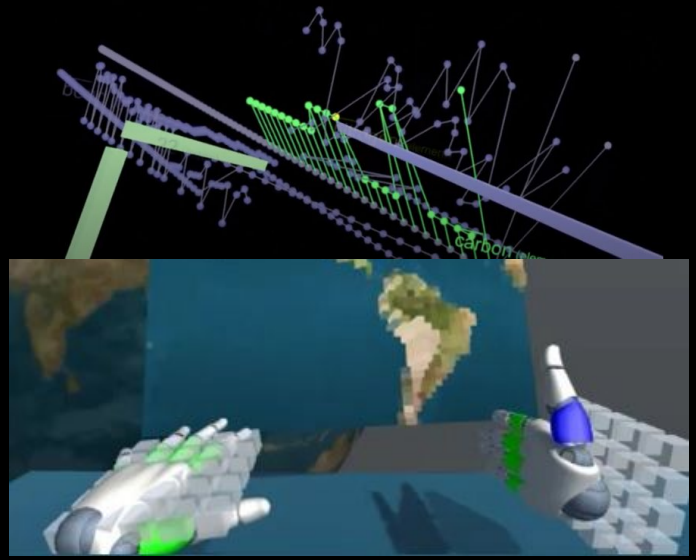


FIG. 10

presentation



interaction

rendering*

*(this talk)

Is that a HyperSpace?

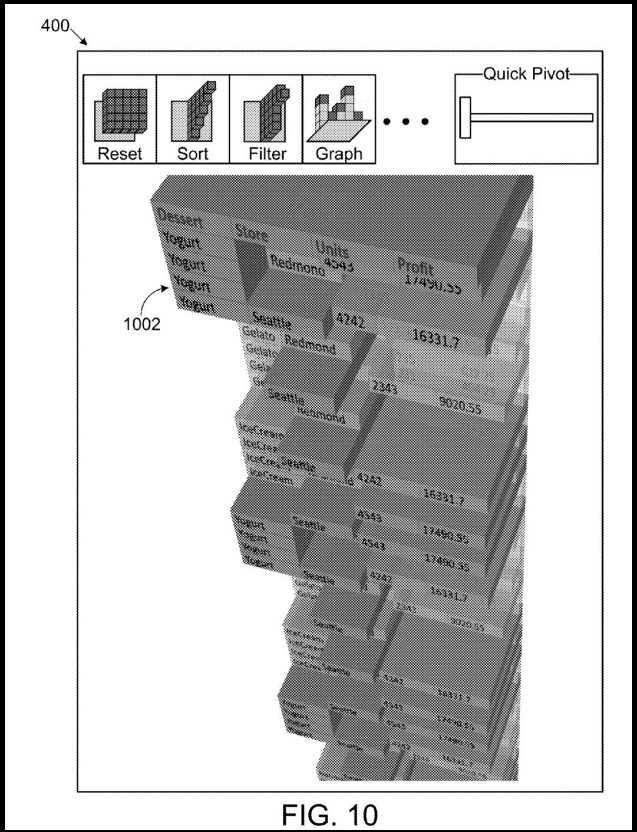


FIG. 10

Charts

{x=domain,y=range,opacity=value(x)<y}

Layout

{ x=_columns, y=_rows, ... }

Rotation

{ x=(_x*0.2 + _y*0.8), ... }

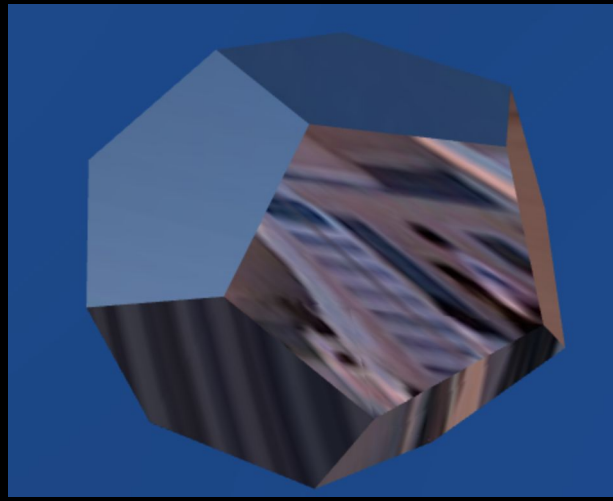
Perspective

(x=(_x/_w), …

Rasterization

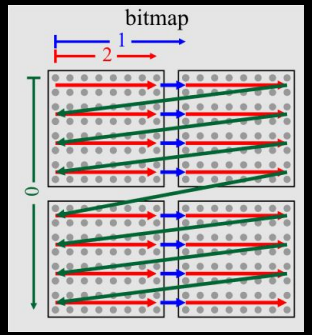
{ r, g, b, depth, x[], y[] }

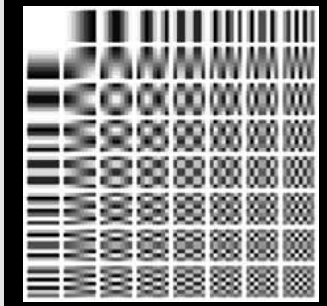
Hyper-Facet

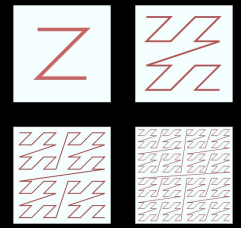


Name - “x”, “green”ness, “cost”, etc.
Range - interval<T> for constants/scales

Appends - data count (e.g. 3 for rgb)
Repeats - array repeats of previous facets (e.g. bitmap width)
Sources - buffers that are indexed
Packing - lookups (div/mod/offset) (calculated within shape)







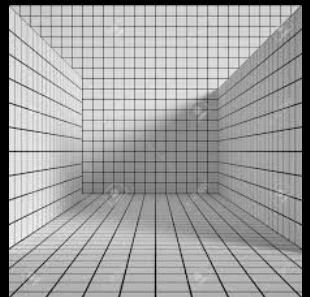
Expression - from input vector to output interval<T> (can use +/*/sine/cosine/etc. Flat O(n))

Hyper-Shapes, -Vectors and -Data

Shape

Facet[]s

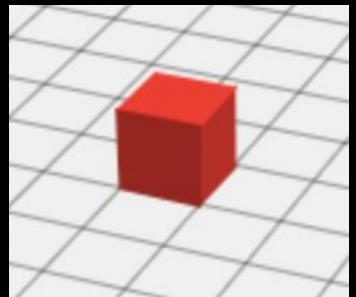
```
Ex.: volShape = [
{name="r",appends=1},
{name="g",appends=1},
{name="b",appends=1},
{name="x",repeats=8},
{name="y",repeats=8}
{name="z",repeats=8}
]
```



Vector

Shape*
Range[]s

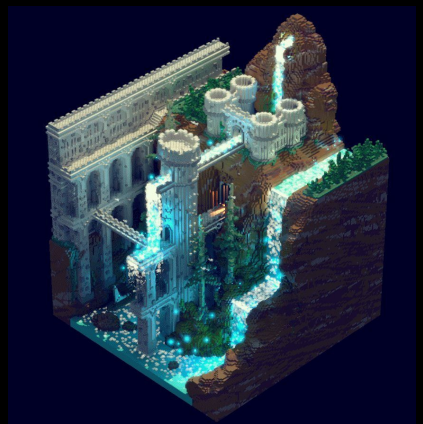
```
{ shape=volShape,
ranges=[
  {1~1},
  {0~0},
  {0~0},
  {4~5},
  {6~7},
  {0~1},
]}
```



Data

Shape*
Buffer

```
{ shape=volShape,
  buffer=[r,g,b,r,...]
}
```



Hyper-Intervals

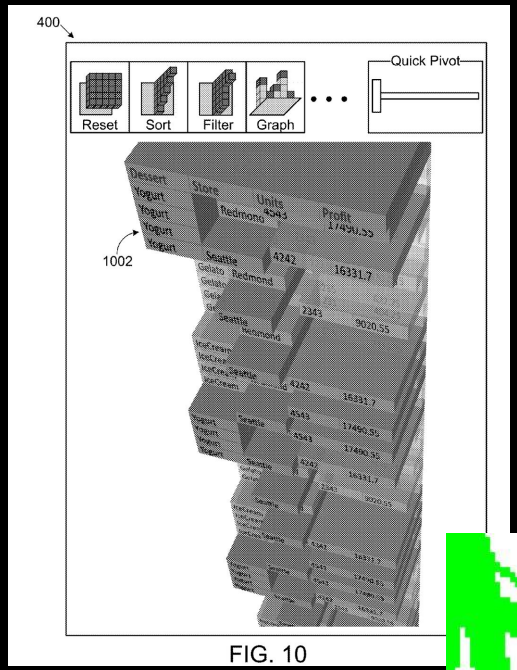
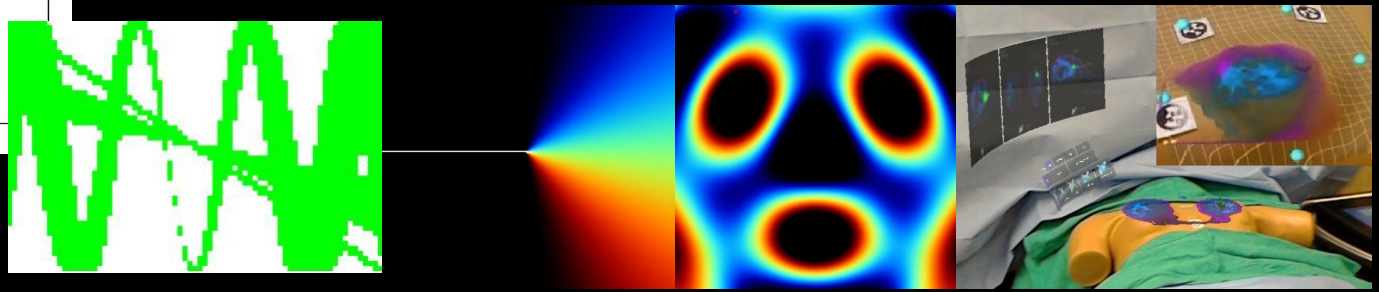


FIG. 10

Interval-arithmetic
- 1D [min and max) bounds, in C/JS/GLSL/etc.
  - See Boost's interval<T>.
- Color as distribution over visible spectrum
- Opacity-intervals as key to volume rendering
- Realtime demos rendering and lighting

E.g. x=0.1~0.2, y=0.8~1.2, time=0~0.016
color=450~500 (blue~cyan)

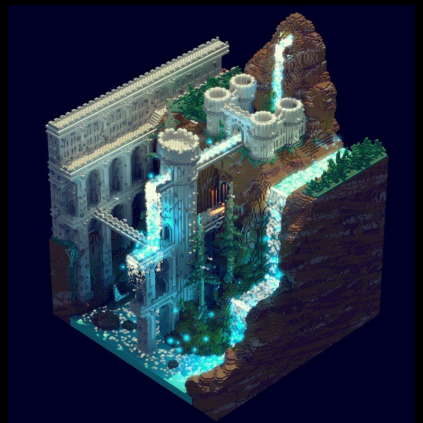


Hyper-Mesh

Data

Shape*
Buffer

```
{ shape=volShape,
  buffer=[r,g,b,r,...]
}
```



Mesh as Indexed Data

VertexData[]
IndexData
MetaData

```
V = { x, y, z, c, vertices[] }
I = { i, j, k, triangles[] }
```

Mesh as Facets

Shape
Buffers[]

```
{shape=[
  x, y, z, vs[],
  [push],
  i∈vs, j∈vs, k∈vs, tris[] ],
buffers=[[0,1,2,0,…],[x,y,z,x…]}
```

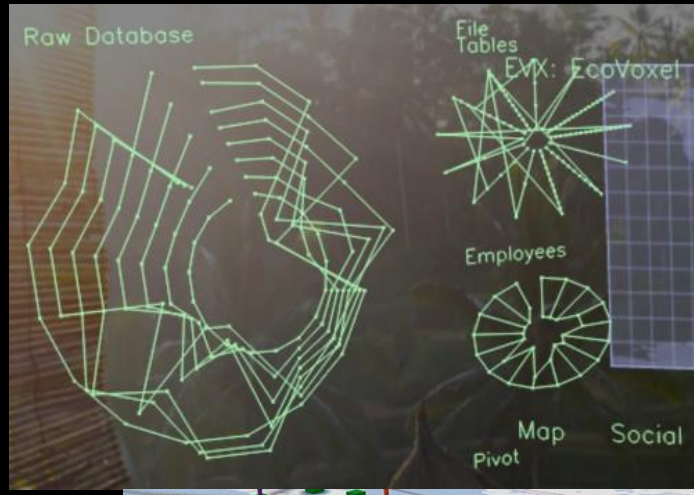
Hyper-Queries

Big-Scene mindset: Query(Shape) => Data

- Convert Shape ranges to SQL
  - Facets/Transforms => SELECT
  - Rasterization => GROUP-BY, ORDER-BY Z
  - Triangles/etc => WHERE

- GPU-based scroll and basic-pivoting
  - Excel 3D Maps 2016 - GPU aggregation (pie charts, heat-maps, etc.)

- Online Maps API
  - Range of lat/lon + resolution required



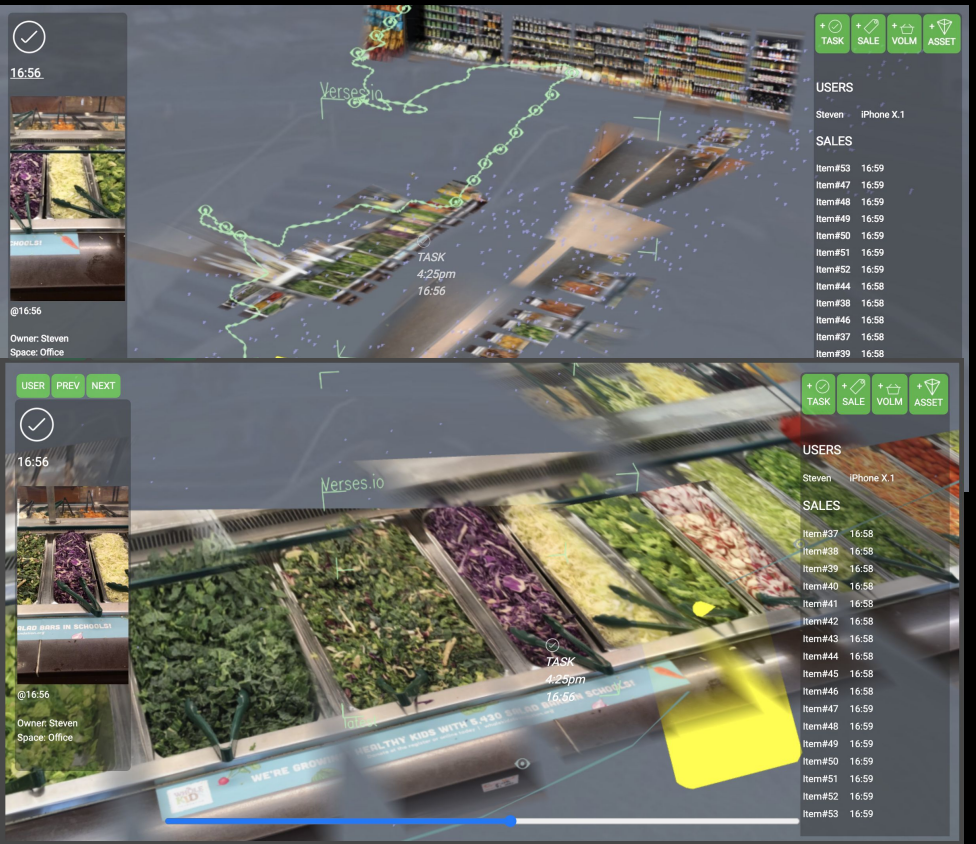
Hyper-Scenes

SceneElement

- **Transform** - positions/**Shape**s sub-content
- **Shaped_Data** - raster/mesh approximation of content
- **Sub_Elements[]** - scene approximation of content

{angle=column, depth=row, radius=value, line=row} {x=lon, y=lat, z=value, line=value=0} (x=lon, y=lat, z=time, line=order_id)

Hyper-Image Reprojection



Combining hyper-spaces

- AR scan of environment
- Path over {x,y,z,time,frame[]}
- Contextual data graphs

Perspective projection

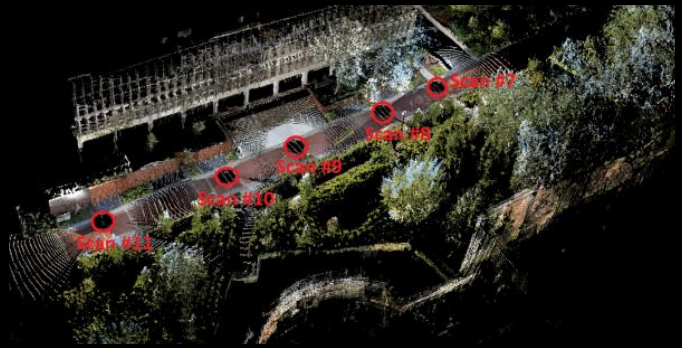
- Texture divide by w in pixel shader
- Image_data = {r,g,b,u[],v[]}
  Points = {x,y,z,vertex[]}
  Triangles = {i,j,k,triangles[]}
  MVP = {x=((_x*m00)+(_y*m01)+...
  Projection = { u=_x/_w, v=_y/_w }

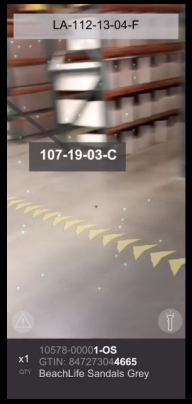
Immersive Cube-Frustum Approximations

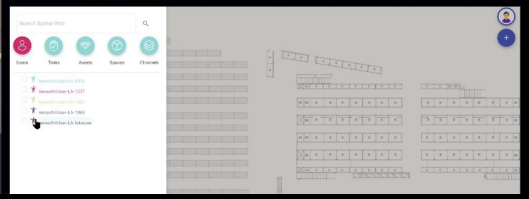
*Google Seurat Announce
(I’m in there)

Recommend alpha-masked frustum approximation of projected cube faces
{r,g,b,z,s,x[],y[],face[],origin[]}
=> {r,g,b,a,u[],v[]}+{x,y,z,u,v,i[]}
tricks and realtime demo:
//alpha=0~0.5 specular=0.5~1.0
//normal~~=ddx/ddy(specular)
//xyz_uv(i)=quad_in_cube_face

Spatial Alignment





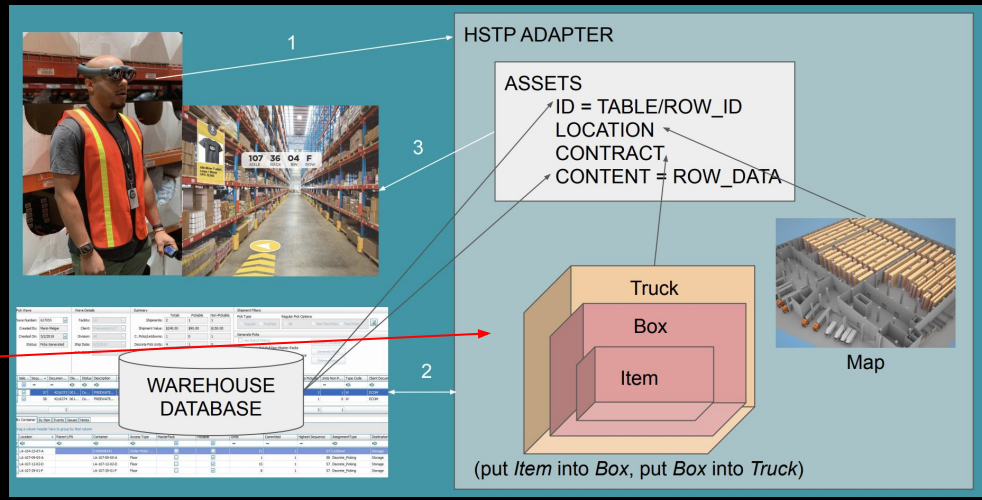


- HSTP's SLAM protocol theory
  - Alignment query format is likely to converge on scene of relatively placed projected images/volumes/features.
  - Query for location of local space "here" relative to known space "building", given hyper-scene of images/features relative to "here".
- Barcodes are the easy way
  - Given location of barcode relative to phone's "here" local space, and location of bar-code in "building", inverse transform of "here" to "building" to localize phone.

Spatial Smart-Contracts

Space as the common language between sensors and business?

- If every task is spatialized, and every thing is tracked… then there is no paperwork, just do the job, and the sensors auto-complete the smart contracts.
- put *Item* into *Box*
  put *Box* into *Truck*

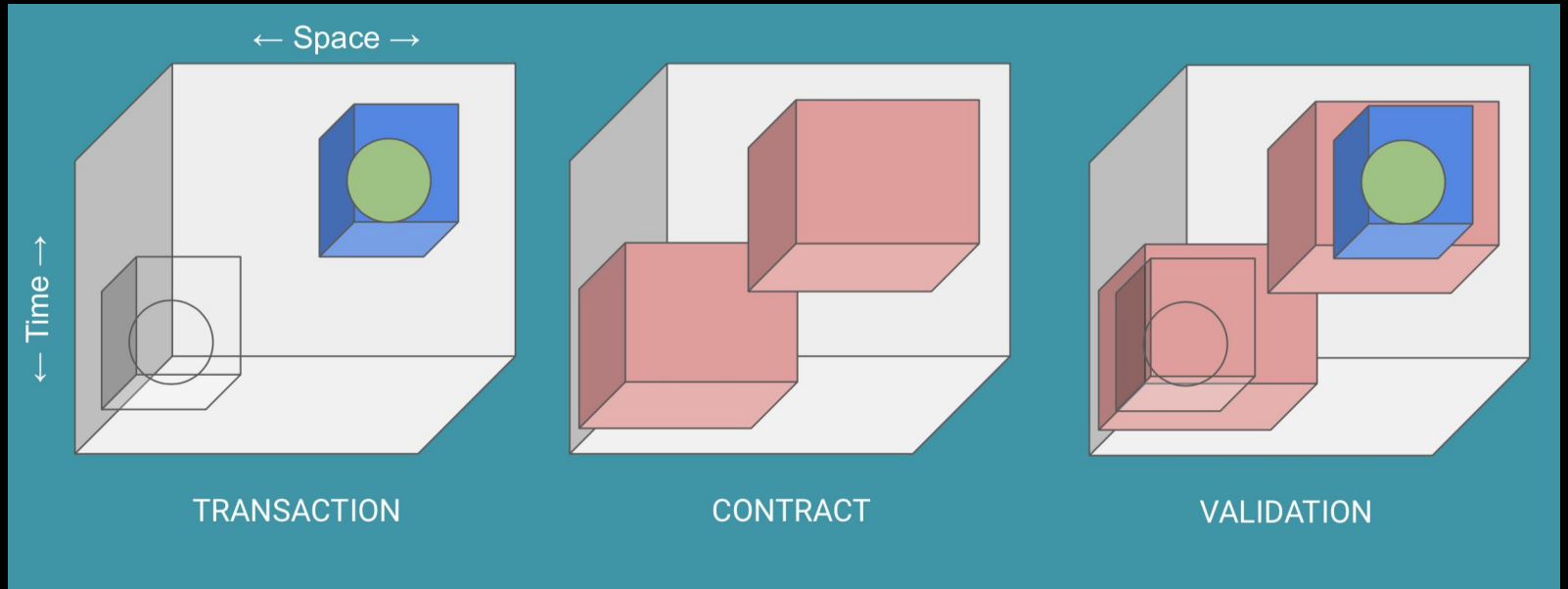


Spatializing Tasks

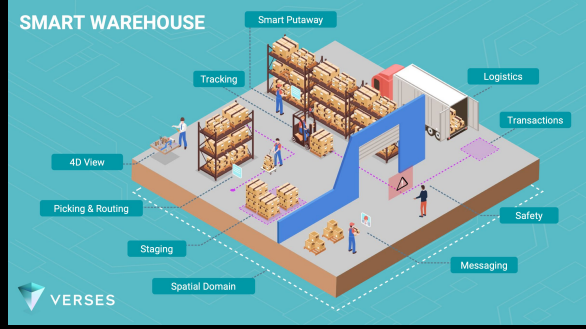
- Most jobs are spatial anyway
- Express metadata as hyperspace intersections
- Express “side-effects” as suggested transactions

Hyper-Transactions and -Contracts

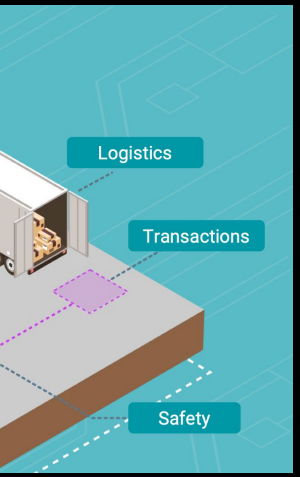
- Transaction is a shape change over block-time
- Contract is the shape of valid block-changes
- Validation is their intersection



Hyper-Contract Performance



- Hyper-Contract-Expression
  - Goal is realtime local validation, of downloaded contracts for: permissions, trade, NFTs/FTs, triggers, etc.
  - Every transaction validated by: target, owner and space (and next owner/space) usually 3 (max of 5).
  - Should evaluate in near O(n) with the number of terms in the contract, BUT does allow intersections.
  - NOT ALLOWED: loops, writes, etc. (not Turing-complete)
  - Side-effects: “suggested changes” can be generated

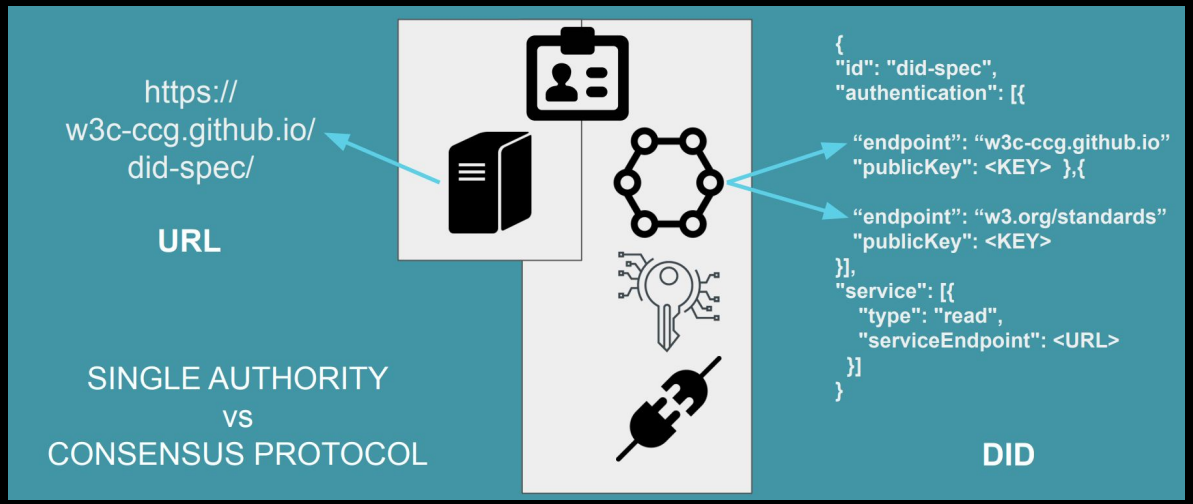


- Hyper-Scene intersection (inside/touching/outside):
  - O(scene_depth * dimensions^2 * shape_expression_length)

Contract avg: O(contract_length)
Contract max: O(contract_length * scene_depth * dimensions^2 * exp_count)

Consensus Network / Decentralized IDs

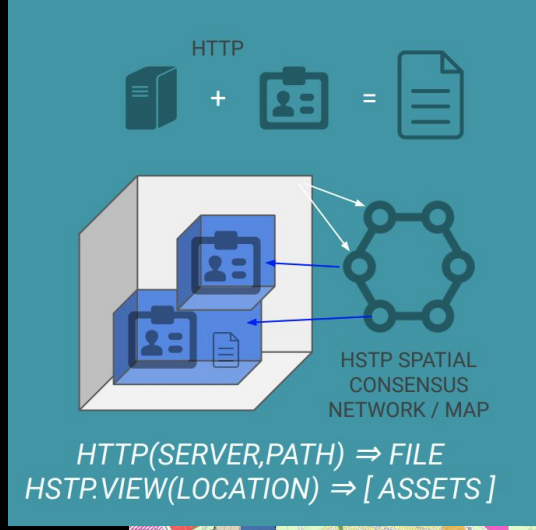
- Each “asset” can have multiple authorities



- W3C DID spec: https://www.w3.org/TR/did-core/ (generally consensus by most/all/any)
- Each ledger contains known assets, and who told them about it, with option to get the authoritative copy, or validate known third-party signed credential (can be ZKP)

Spatial Network

- Spatial Network Routing
  - SendSpatialMessage(sender, range, message)
  - Location as ID is actually quite common
    - SQL update, physical Post-Office, Internet backbone, Physics sim
- Hyper-Scene Hierarchy
  - Each asset has a scene transform parent reference.
  - Common parent/grandparent/etc. required for intersection calculation.
  - Bounds intersection: O(parents * dimensions^2)
- Spatial Domain Registry?
  - Will land rights and most vehicles be digitized and smart execute as their location updates? … yes
  - *(image of me and ESRI founder Jack Dangermond discussing spatial smart contracts)



Ethics as Spatial Contracts

Physical Boundaries		Clarity of Mandate
Privacy Boundaries		Where/What, not Who
Social Boundaries		Social Boundaries
Economic Boundaries		Spacial Currencies

From *“How to Express Ethics as Spatial Smart-Contracts”*,
talk I gave at Microsoft Research in response to their HoloLens military contract.

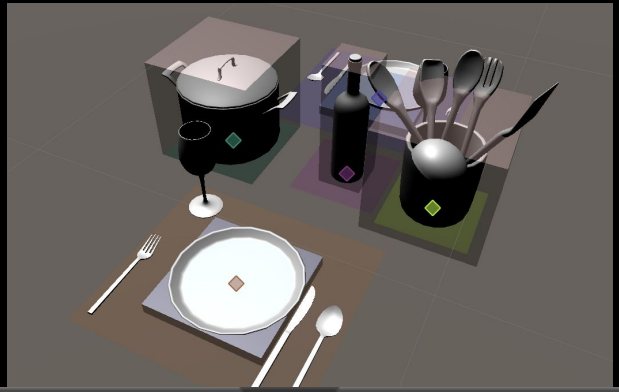
Contextualizing Scenarios

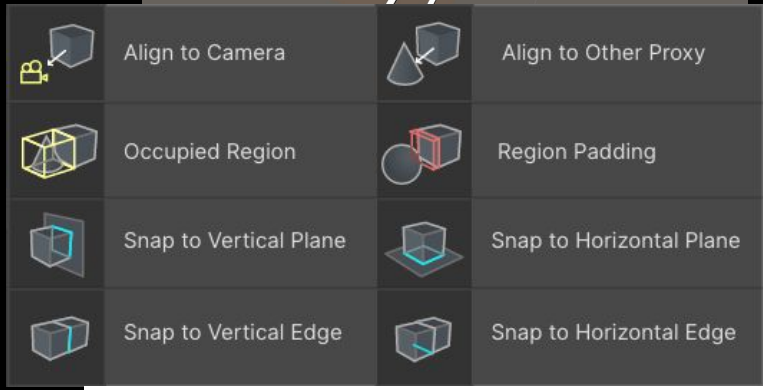
How to create adaptable spatial scenarios?

- Solution: model the positive and negative spaces (occupied/empty)
- See Unity MARS “Proxy Forces” API
- And of course iterate lots (see Unity MARS for in-editor sim)

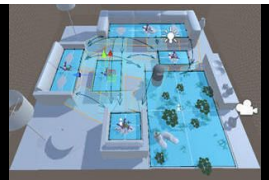
Solving complex alignments?

- Root solving is costly in this case
- Iterate: sample environment over local space to estimate gradient towards root





$$min(Q(T)) \mid Q(T) = \int_{v}^{B} \left| \overline{E}(v * T) - \overline{P}(v) \right| dv$$



Hyper-State Machines

- How to describe the shape of a text file?
  - Row and column start at 0, and change based on what the next read character is.

```
Raw File: {
  char,
  index[] }
```

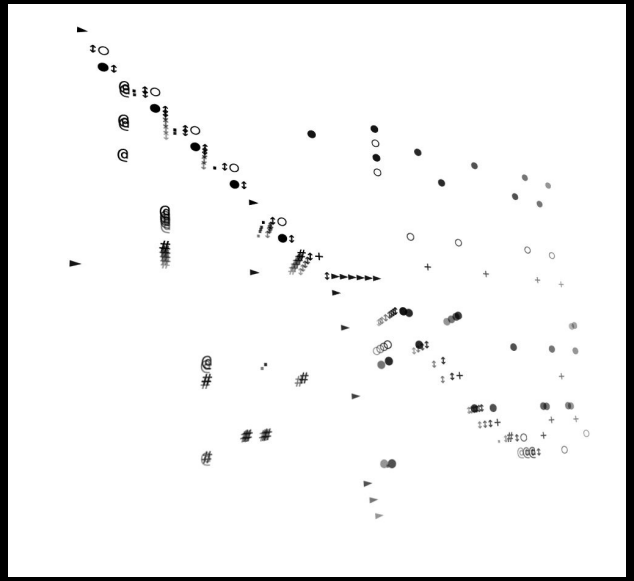
```
Text File: { char,
  row=((char=='\n')?(_row+1):_row),
  col=((char=='\n')?0:(_col+1)),
  word=(isSpecial(char)?(_word+1):_word),
  time[] }
```

```
function helloWorld(app) {
    app.app_in.subscribe((input) => {
        app.app_out.copy(input);
        app.app_out.frameStep();
    });
}
```

Turing-machines are examples of this

- “a computation” is really a projection of initial state into the shape of a Turing machine…
- Visual sandbox & safe execution environment…

Lewdo Kernel



legend

3Dtime
↕addresses ↔values

○read ●write .dot
#number @object
+add ►call ~free

(deeply in progress)... fixed size micro kernel for updating a thread that uses instructions, stack, and heap.
{ letter=wiring, x=values[], y=addresses[], z=times[] }

Challenges

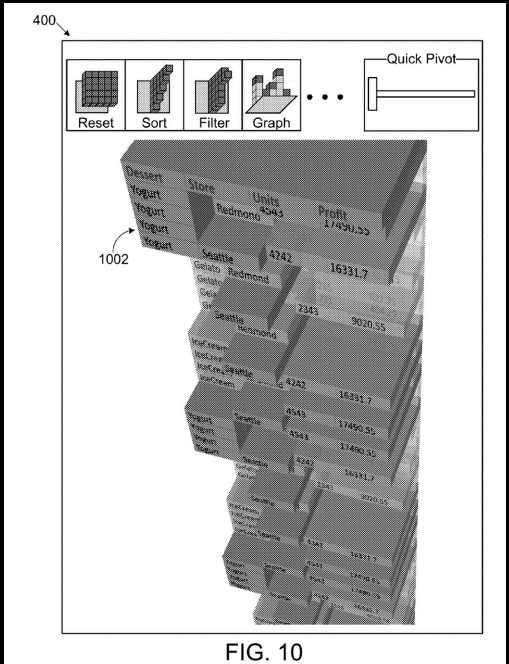
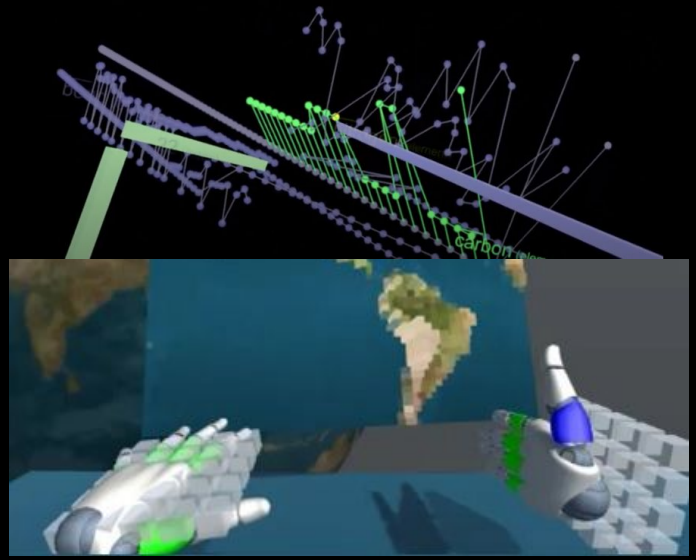


FIG. 10

presentation



interaction

rendering*

*(this talk)


Thank you
-Lewey G.