Advanced data visualization using the Forge viewer

Kean Walmsley

Platform Architect, Autodesk Research

About the speaker

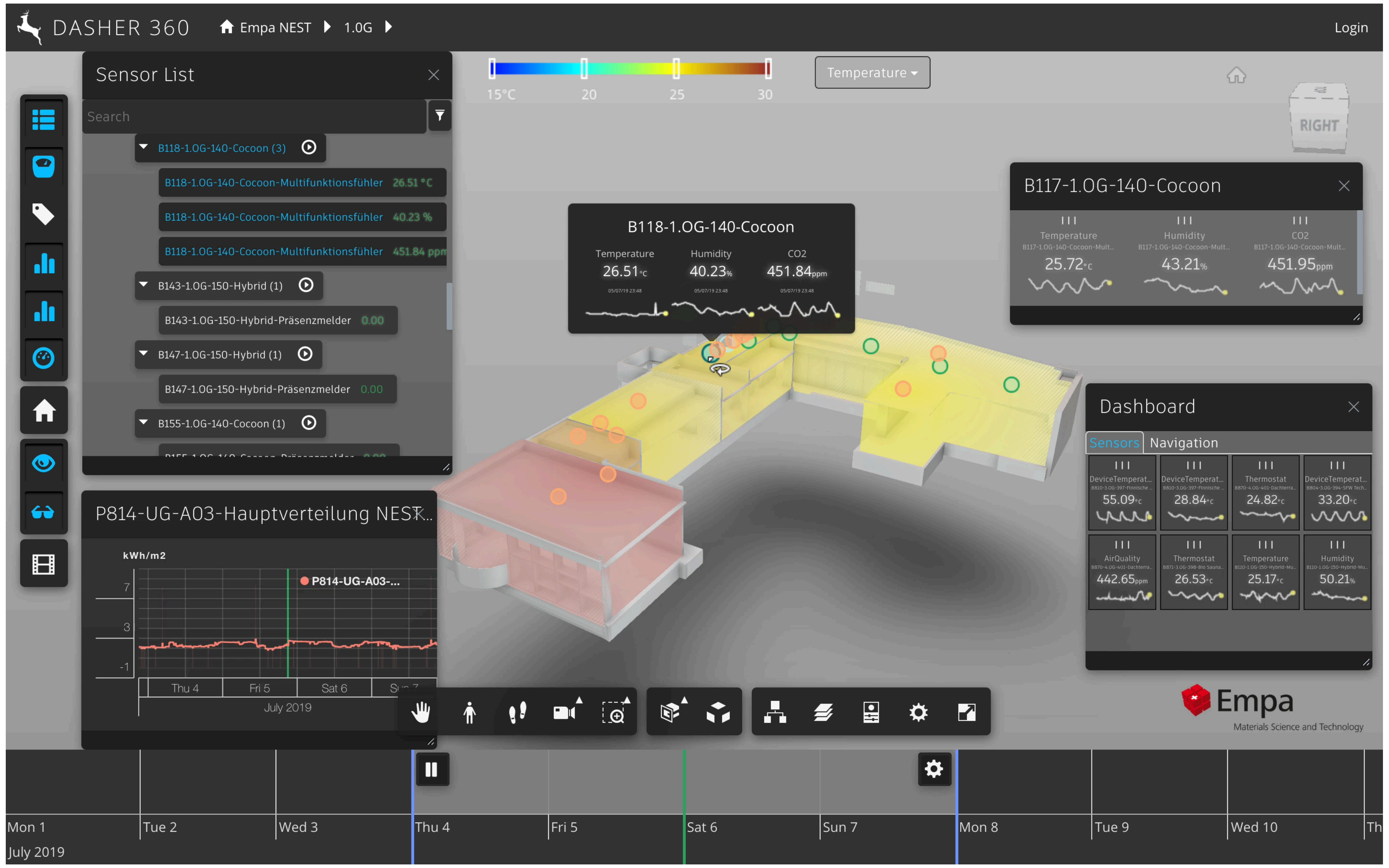
Kean Walmsley

1995-2012	ADN	1995-1998	UK
2012-2016	AutoCAD	1998-2000	Switzerland
2016- …	Research	2000-2003	USA
		2003-2005	India
		2006- …	Switzerland

2006- …	Through the Interface

Agenda

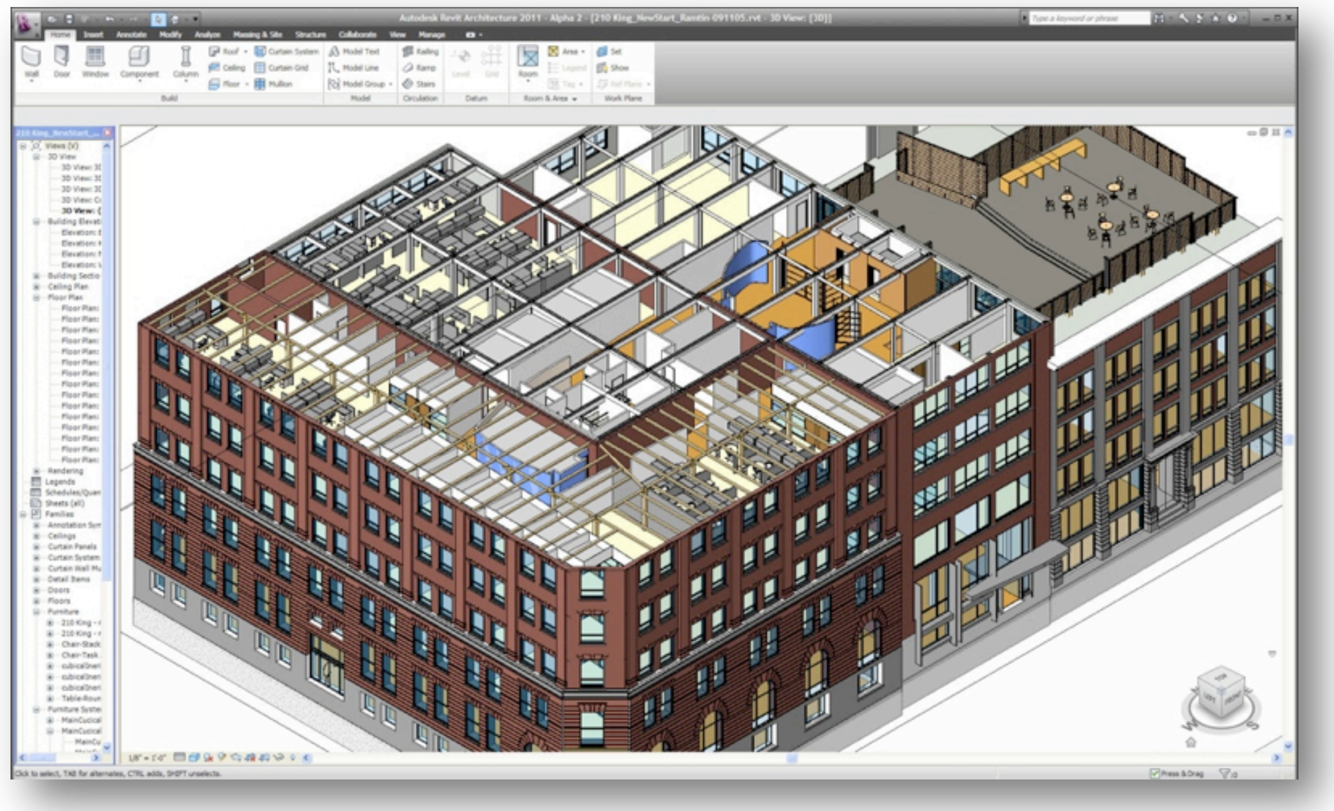
- History of Project Dasher
- How Dasher 360 uses Forge
- External libraries
- Using Forge to visualize IoT data
  - Focus on 3D visualization, not data retrieval
- Useful Forge examples
- What's next for Dasher?



Introduction to Project Dasher

History of Project Dasher

A research project to develop visualization and analytics tools for operations data in the context of BIM

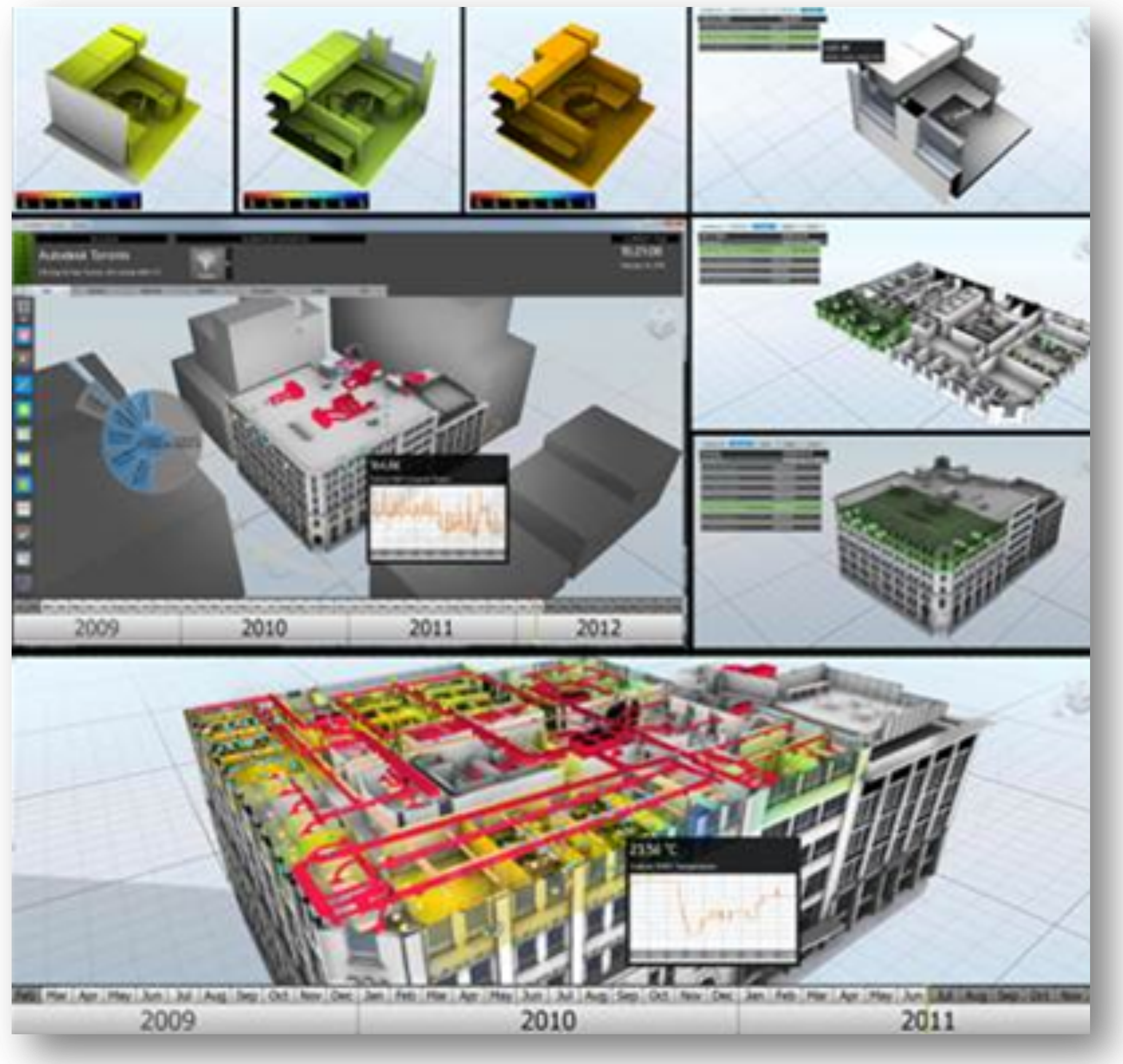


As-Built BIM
Highly detailed AEC models

+

Building Data
Building management systems (BMS) and IoT enabled sensors for operations and management data collection

=

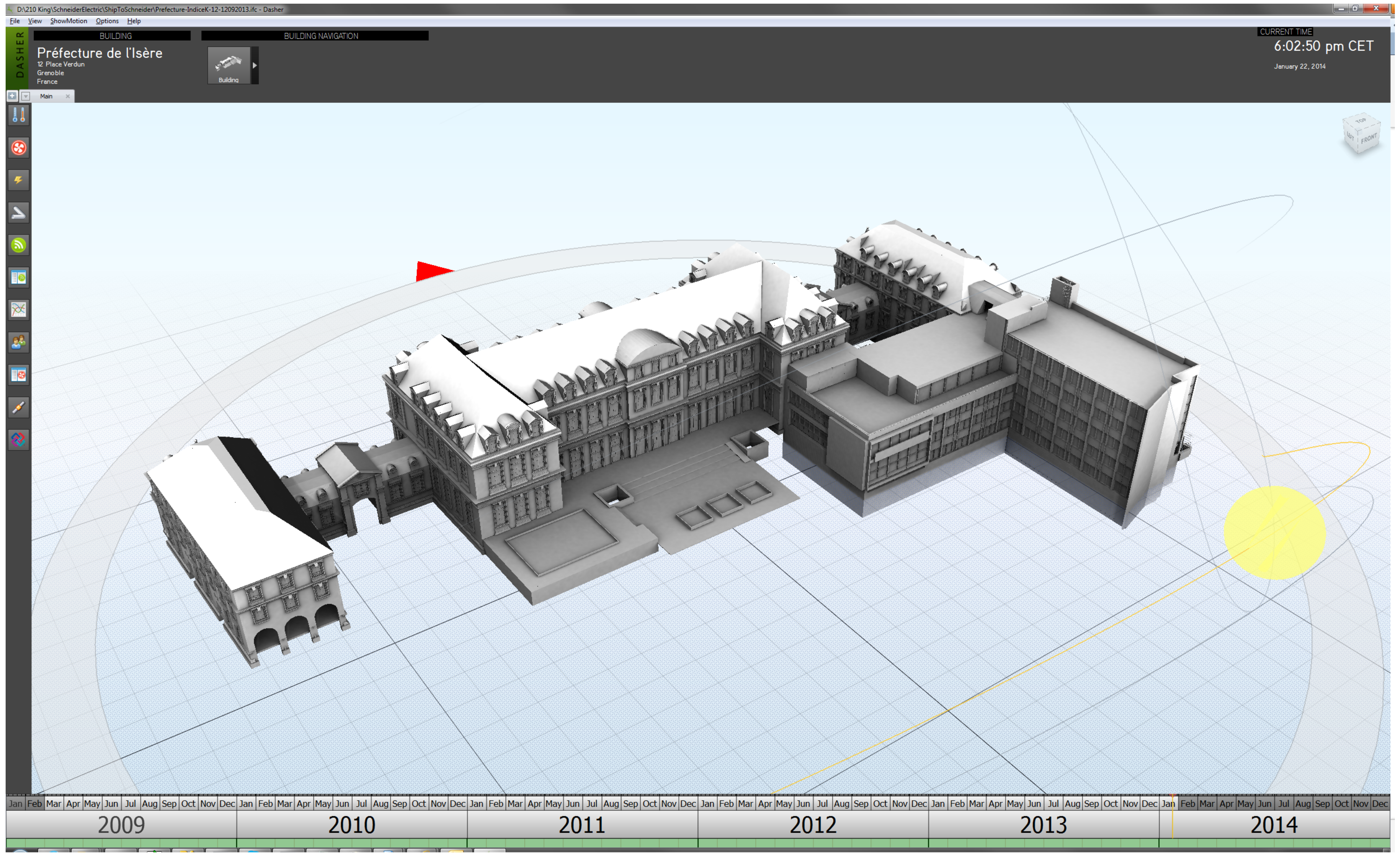


Project Dasher
A visualization tool to help customers understand their data in context of the 3D model and debug operational issues

2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 20

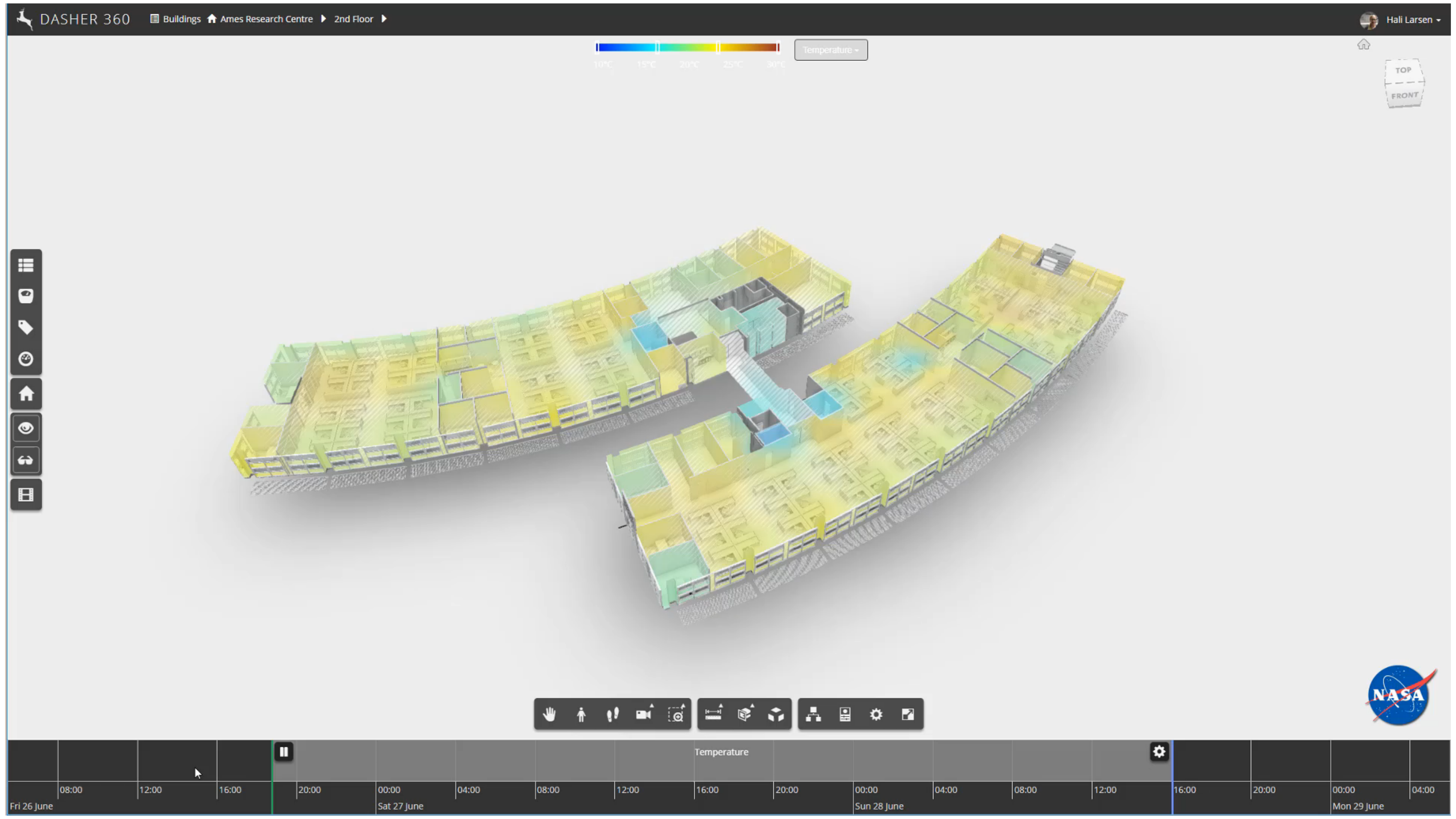
History of Project Dasher

History of Project Dasher



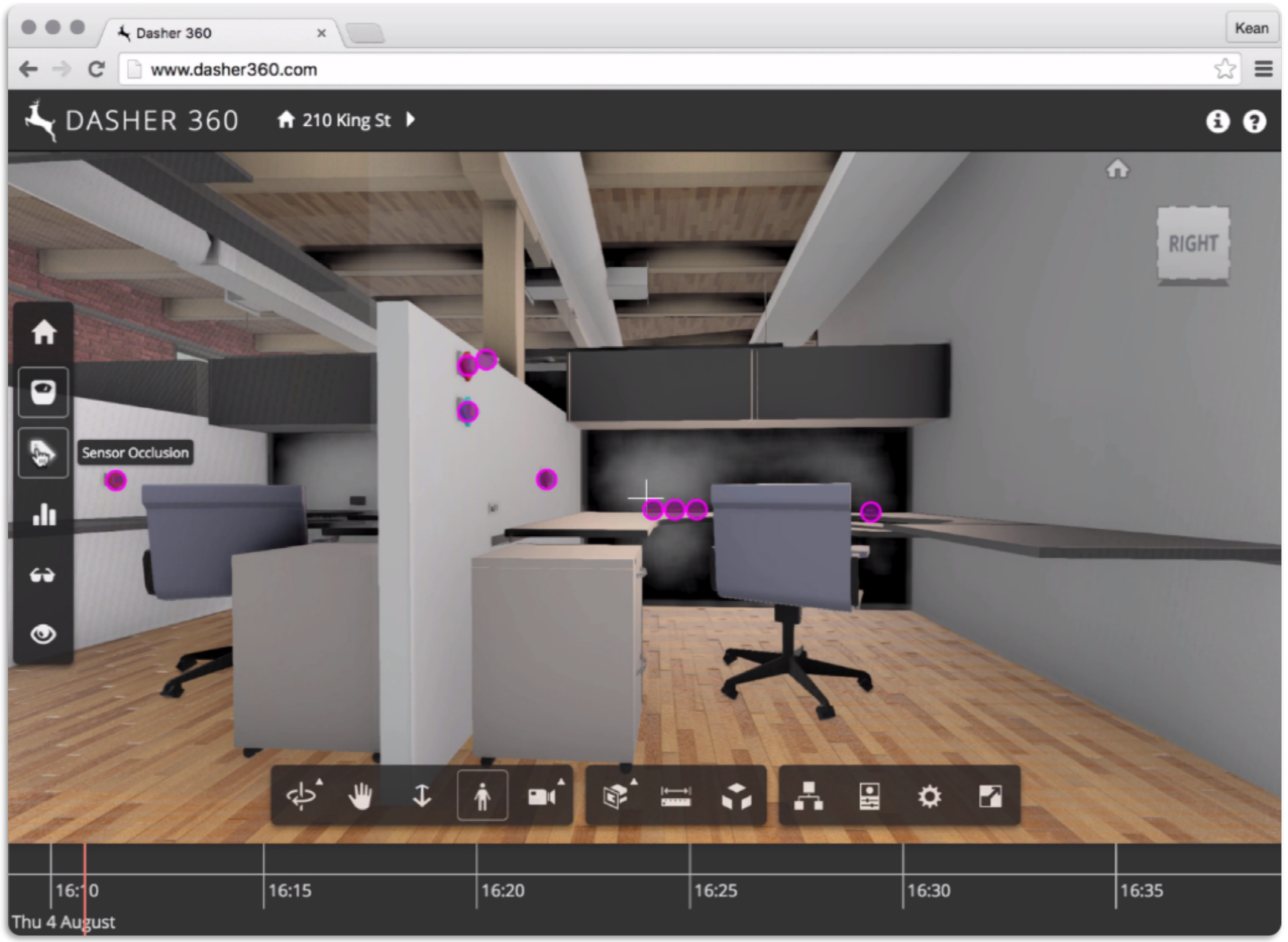
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
20

History of Project Dasher



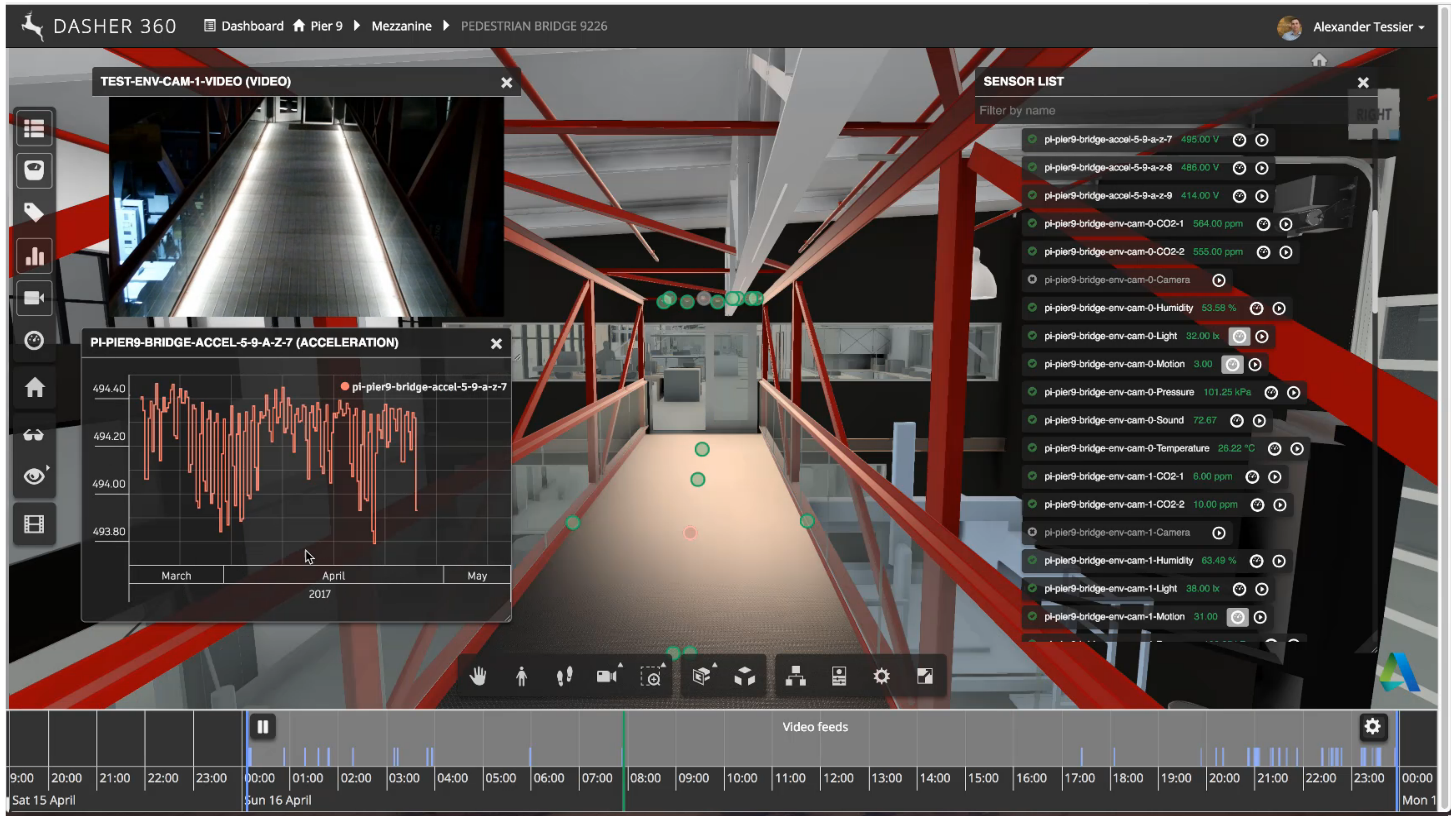
2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 20

History of Project Dasher



2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
20

History of Project Dasher



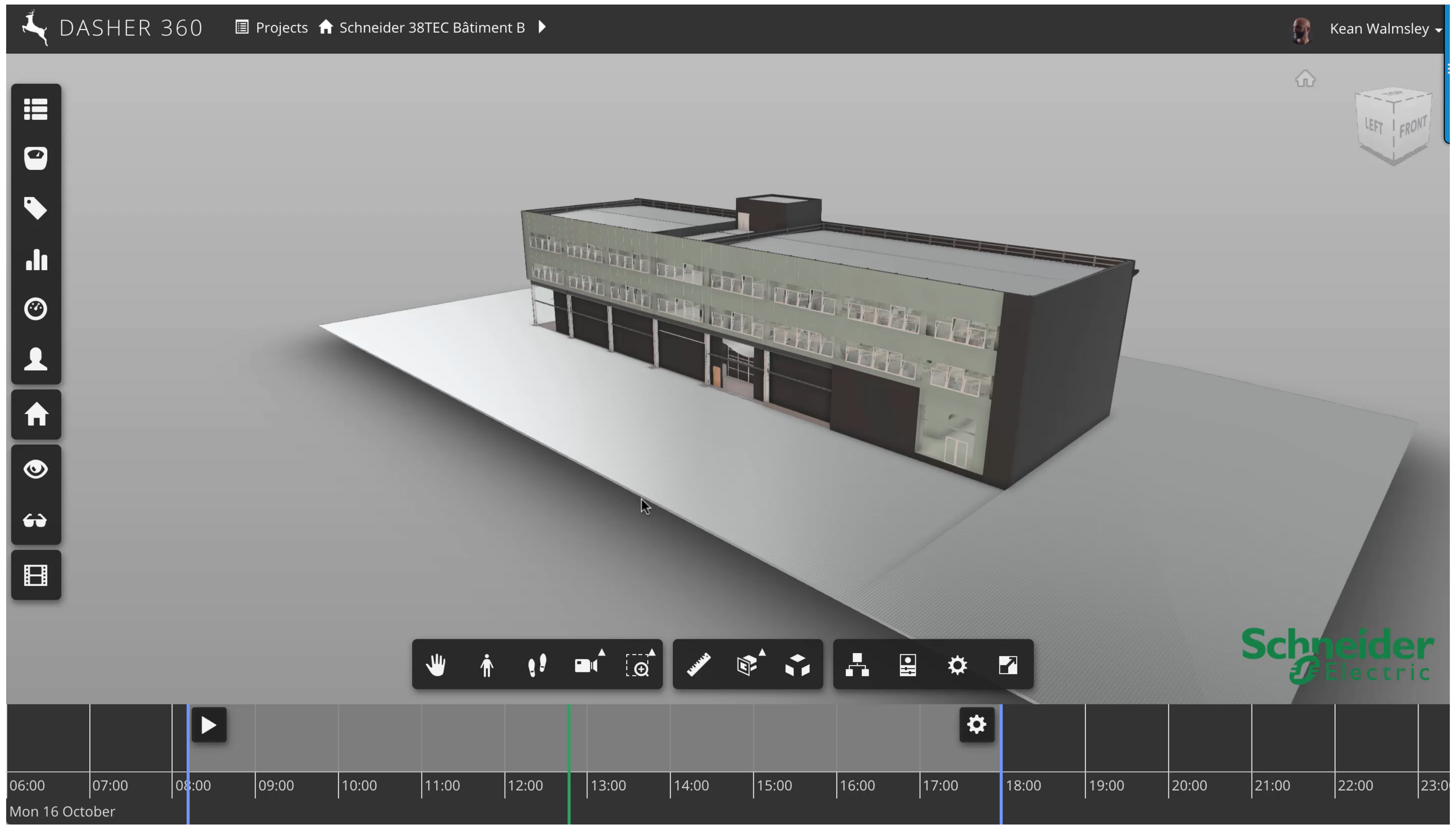
SENSOR LIST

Filter by name

pi-pier9-bridge-accel-5-9-a-z-7	495.00 V
pi-pier9-bridge-accel-5-9-a-z-8	486.00 V
pi-pier9-bridge-accel-5-9-a-z-9	414.00 V
pi-pier9-bridge-env-cam-0-CO2-1	564.00 ppm
pi-pier9-bridge-env-cam-0-CO2-2	555.00 ppm
pi-pier9-bridge-env-cam-0-Camera	
pi-pier9-bridge-env-cam-0-Humidity	53.58 %
pi-pier9-bridge-env-cam-0-Light	32.00 lx
pi-pier9-bridge-env-cam-0-Motion	3.00
pi-pier9-bridge-env-cam-0-Pressure	101.25 kPa
pi-pier9-bridge-env-cam-0-Sound	72.67
pi-pier9-bridge-env-cam-0-Temperature	26.22 °C
pi-pier9-bridge-env-cam-1-CO2-1	6.00 ppm
pi-pier9-bridge-env-cam-1-CO2-2	10.00 ppm
pi-pier9-bridge-env-cam-1-Camera	
pi-pier9-bridge-env-cam-1-Humidity	63.49 %
pi-pier9-bridge-env-cam-1-Light	38.00 lx
pi-pier9-bridge-env-cam-1-Motion	31.00

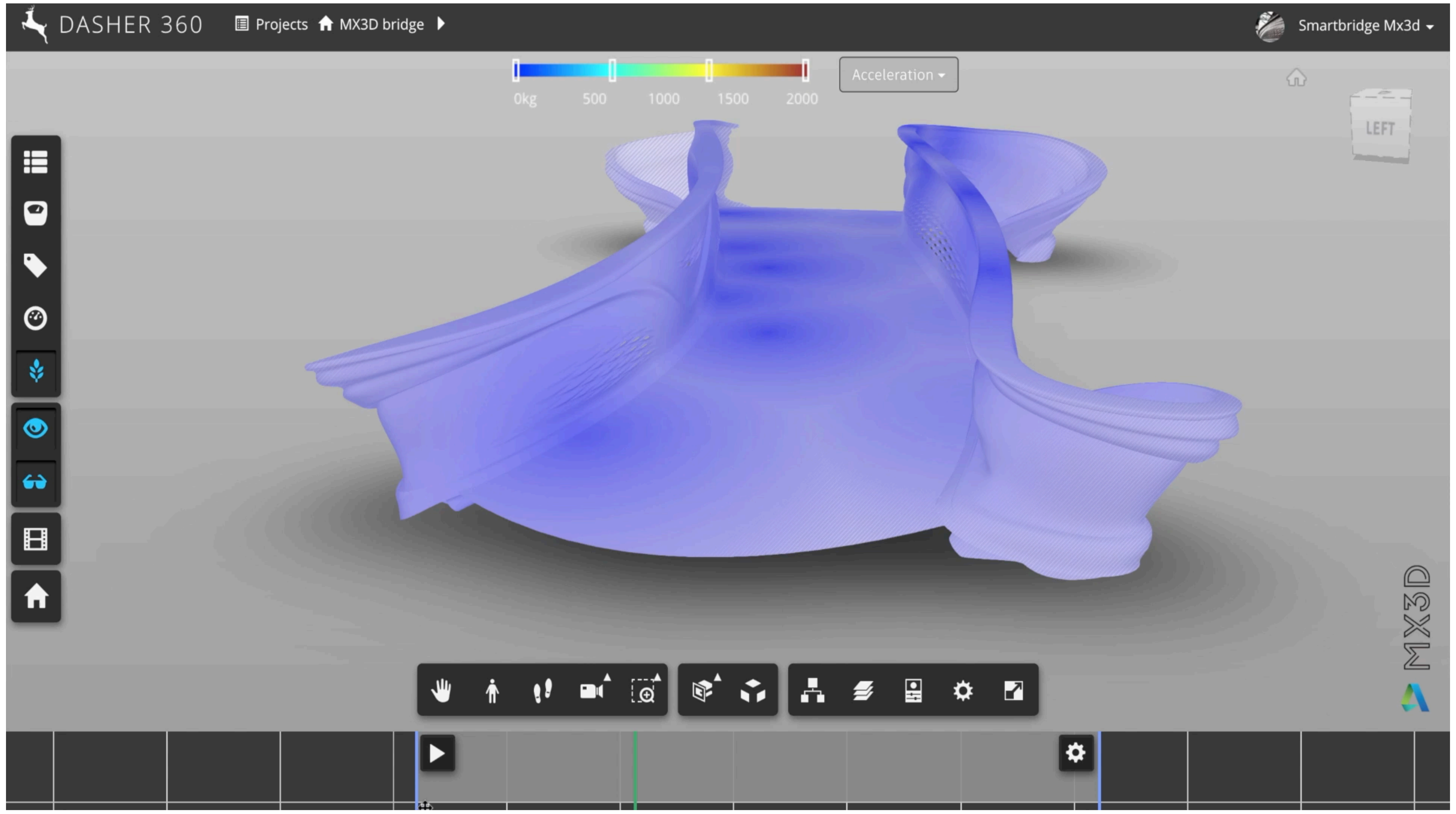
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
20

History of Project Dasher



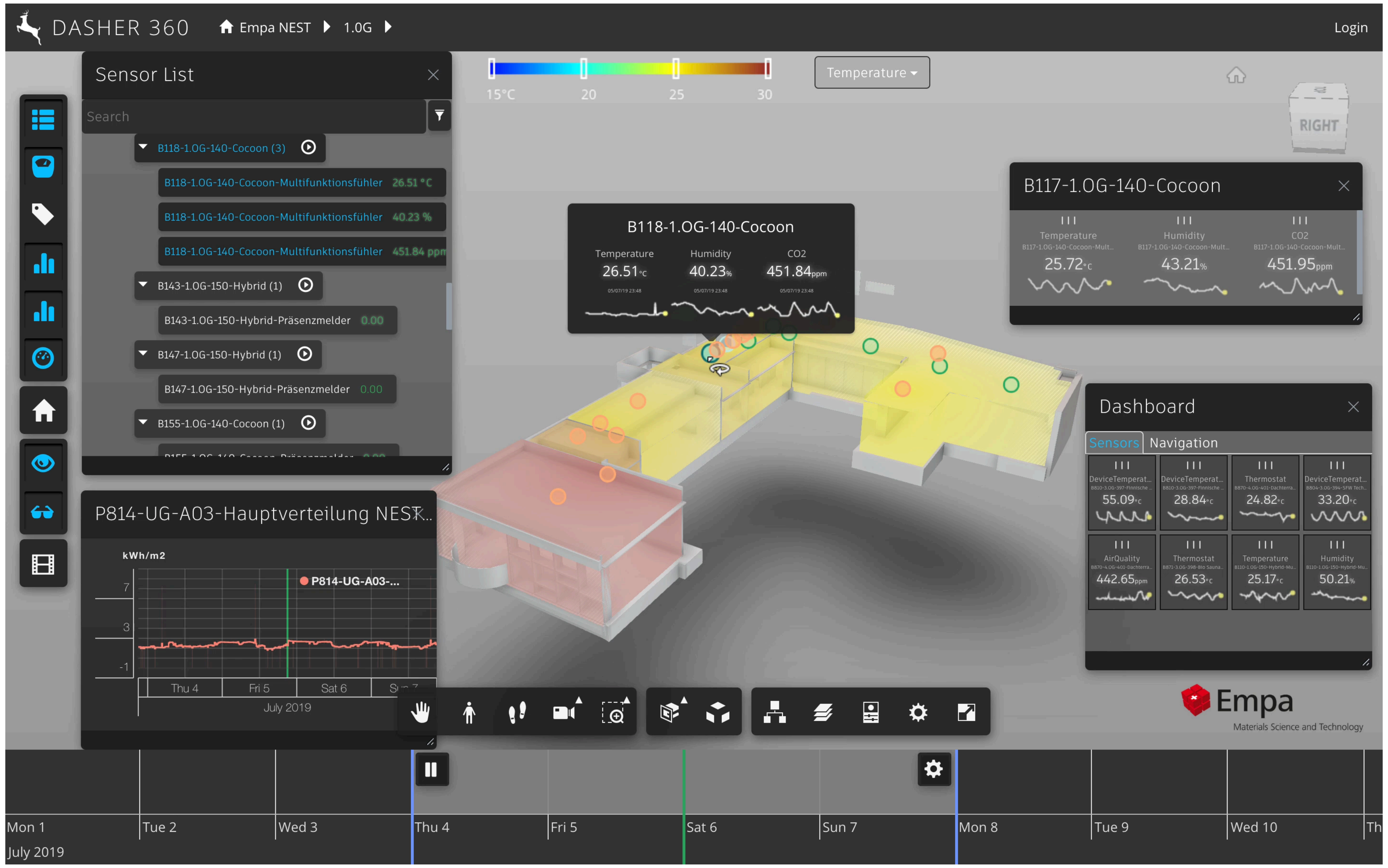
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
20

History of Project Dasher



2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 20

History of Project Dasher



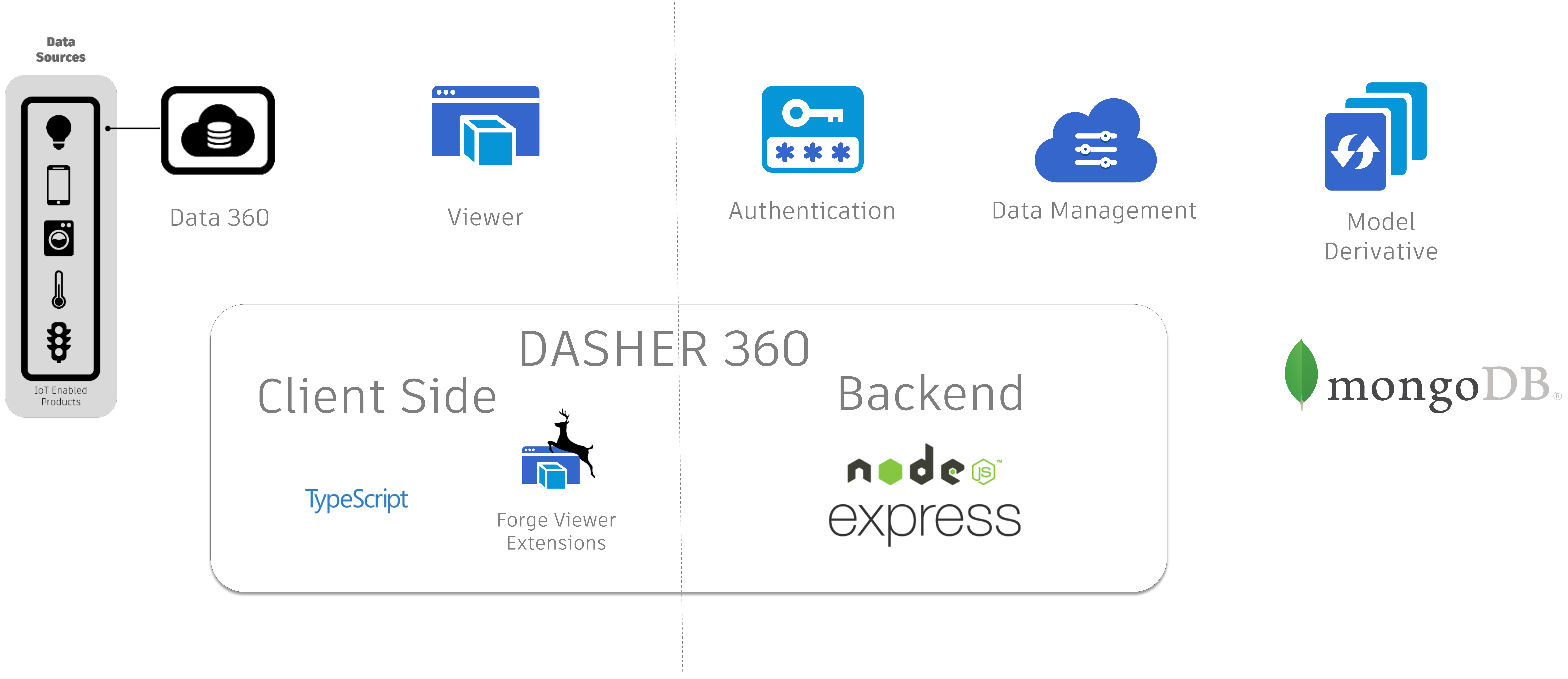
How Dasher 360 Uses Forge

Forge Services

- Authentication
  - Leverage Forge authentication for private access
- Data Management API
  - Access models in BIM 360 Team and Docs
- Model Derivative API
  - Translate models for public sharing

Dasher's Forge Usage

AUTODESK® FORGE



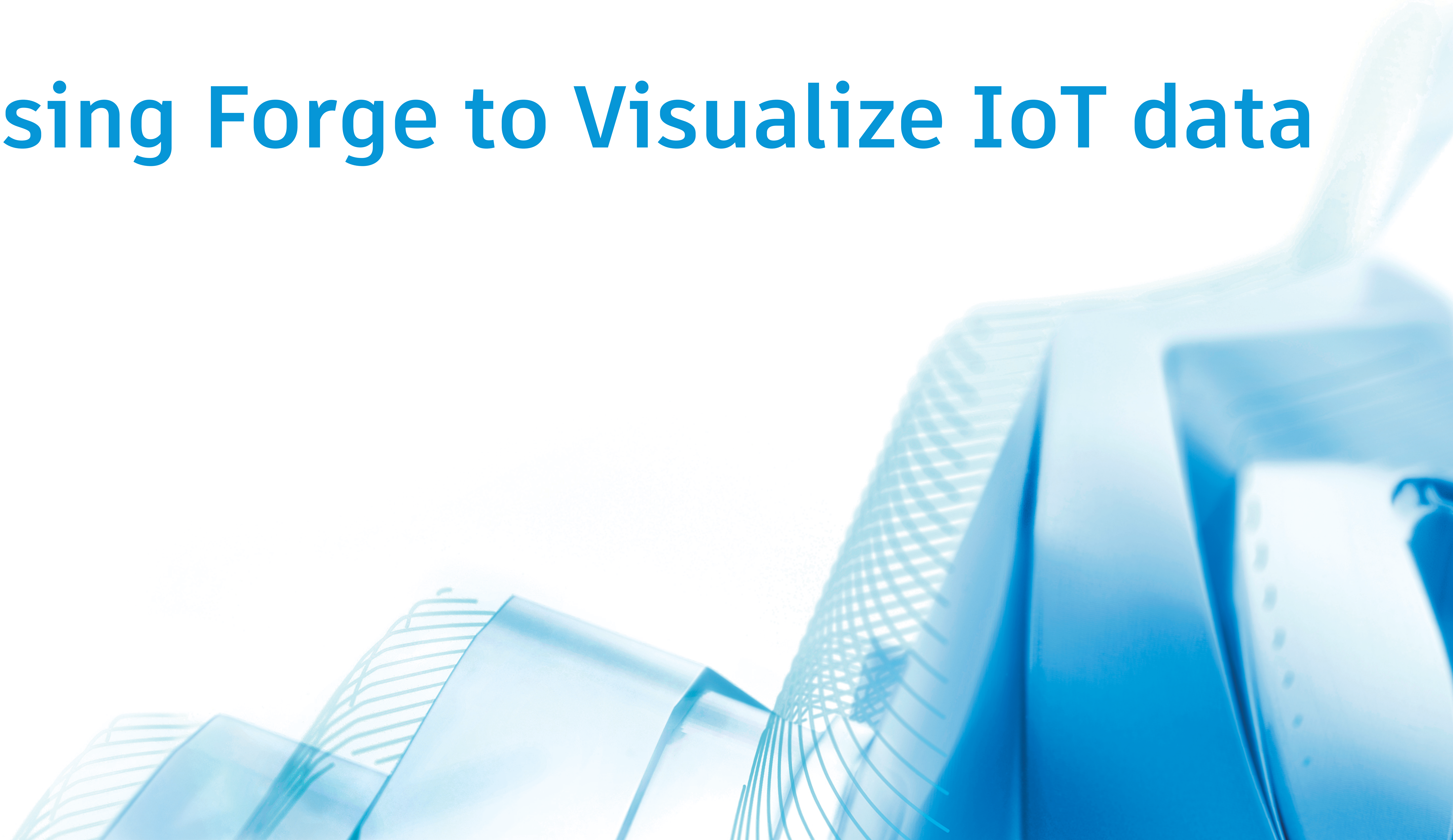
External Libraries

• Bootstrap	github.com/twbs/bootstrap
• Fancytree	github.com/mar10/fancytree
• Tooltipster	github.com/iamceege/tooltipster
• Sparkline	github.com/gwatts/jquery.sparkline
• Gridstack	github.com/gridstack/gridstack.js
• Vis.js	github.com/visjs
• MeshLine	github.com/spite/THREE.MeshLine (need the fork for R71)

Forge Viewer

- Dasher 360 uses **lots** of Forge viewer features
- The most critical ones for integrating IoT data in a 3D context are:
  - **Overlays** provide the ability to add custom scene data via THREE.js
  - **Custom shaders** are used extensively (some added via overlays)
    - Render a point cloud to show sensor positions and support hovering
    - Display animated heatmaps in volumes or on objects
    - Determine whether a location is inside a particular room

Using Forge to Visualize IoT data



Overlays

- We use Overlays to add custom graphics into the Forge viewer scene
    - Skeletons
    - Streamlines
    - Robots
    - Custom selection highlighting
    - X-Rayed objects
- Works best when material's `depthTest` flag set to false

Overlays

- There's now a public API allowing you to do this, via `viewer.overlays`

```
export class OverlayManager {
  addScene(name: string): boolean;
  removeScene(name: string): void;
  clearScene(name: string): void;
  hasScene(name: string): boolean;
  addMesh(mesh: THREE.Mesh|THREE.Mesh[], name: string): boolean;
  removeMesh(mesh: THREE.Mesh|THREE.Mesh[], name: string): boolean;
  hasMesh(mesh: THREE.Mesh, name: string): boolean;
}
```

Overlays

- Create a scene with a custom name via `viewer.overlays.addScene()`
  - e.g. ‘DasherSkeletons’
- Add the objects you need via `viewer.overlays.addMesh()`
  - Doesn’t strictly need to be a mesh
    - We also use this for PointCloud objects, for instance
- Hide using `viewer.overlays.removeScene()`

Overlay Demo

- Floor selection
- Skeletons
- Streamlines
- Robots
- X-Ray

Custom Shaders

- Shaders use the GPU to render 2D images (such as for the screen buffer)
  - They’re quick: they get compiled and then run on the GPU
  - They’re highly parallel (one reason they’re quick)
  - The code is written in GLSL: this is often in plain text in the browser
    - It’s a low-level, C-like language, and can be tricky
  - Get started with this excellent resource: thebookofshaders.com

Custom Shaders

- A shader is made up of 2 GLSL programs
  - **Vertex shader** transforms each vertex's 3D position in virtual space to its 2D screen coordinate (as well as a depth value for the Z-buffer)
  - **Fragment shader** decides the color to paint each pixel

Custom Shaders – Sensor Dots

- For scalability, sensor locations are managed via PointClouds
  - Rendered using a custom shader material, e.g.

```
material = new THREE.ShaderMaterial({
  uniforms: uniforms,
  vertexShader: vertexshader,
  fragmentShader: fragmentshader,
  blending: THREE.NormalBlending,
  transparent: true
});
```

Vertex shader

```
attribute vec3 outerColor;
attribute vec3 innerColor;
attribute float visible;
attribute vec3 nDir;
attribute float offset;
attribute float hovered;
attribute float type;
uniform vec2 vpSize;
varying vec3 oColor;
varying vec3 iColor;
varying float vis;
varying float nodeType;

float pixelWidthRatio;

void main() {
  pixelWidthRatio = 1.0 / (vpSize.x * projectionMatrix[0][0]);
  oColor = outerColor;
  iColor = innerColor;

  mat4 mat = projectionMatrix * modelViewMatrix;
  vec4 originalPosition = mat * vec4(position, 1.0);
  float pixelWidth = originalPosition.w * pixelWidthRatio;
  float resRatio = 0.5;
  vec3 off = vec3(0.0,0.0,0.0);
  if (length(nDir) > 0.0) {
    off = normalize(nDir) * resRatio * (` + size + ` * 0.5 * pixelWidth + offset);
  }
  if (hovered > 0.0) {
    gl_PointSize = ` + size + ` * 1.2;
  } else {
    gl_PointSize = ` + size + `;
  }
  nodeType = type;
  vec3 offsetPt = position + off;
  vis = visible;
  gl_Position = mat * vec4(offsetPt, 1.0);
}
```

Fragment shader

```
uniform sampler2D texture;
uniform sampler2D texture2;
varying vec3 oColor;
varying vec3 iColor;
varying float vis;
varying float nodeType;
void main() {
  if (vis < 0.5) discard;
  vec4 opacity;
  if (nodeType > 0.0) {
    opacity = texture2D(texture2, gl_PointCoord);
  } else {
    opacity = texture2D(texture, gl_PointCoord);
  }
  vec3 c = mix(oColor, iColor, opacity.x);
  float a = (1.0-(0.4*opacity.x)) * opacity.w;
  vec4 color = vec4(c, a);
  if (color.w < 0.5) discard;
  gl_FragColor = color;
}
```

Custom Shaders – Sensor Hovering

- We used to raycast into the PointCloud(s) to determine cursor hits
  - Fire a ray from the cursor location in the camera direction, etc.
  - Expensive, imprecise and difficult to maintain
- We now draw a bitmap of the screen and check the pixel at the cursor pos
  - Sensor ID is encoded in the pixel color via a float texture
    - This bitmap is typically never shown, so the visual color isn’t important
  - Efficient, reliable and supports custom point shapes

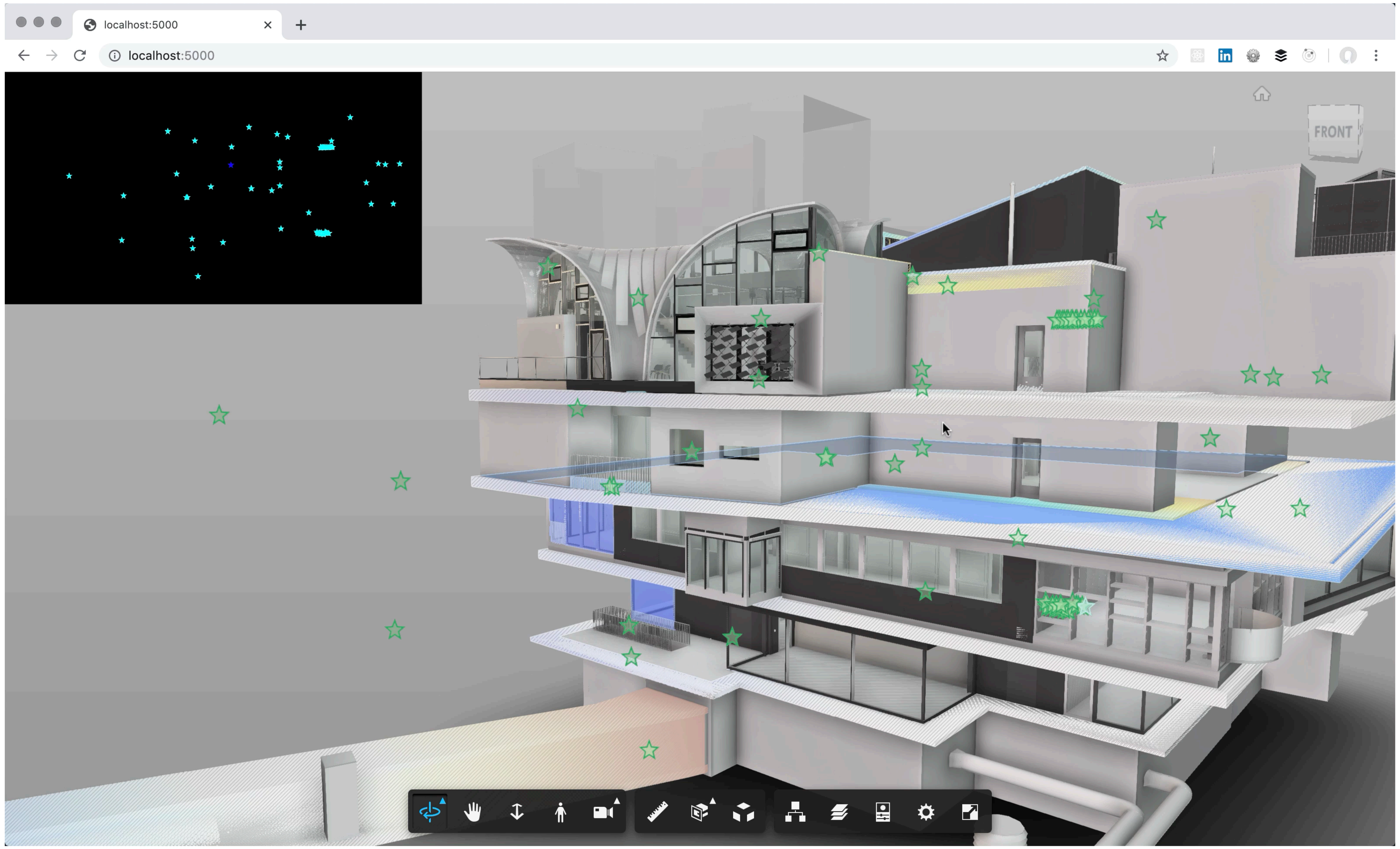
Vertex shader

```
attribute vec3 outerColor;
attribute vec3 innerColor;
attribute float visible;
attribute vec3 nDir;
attribute float offset;
attribute float hovered;
attribute float type;
uniform vec2 vpSize;
varying vec3 oColor;
varying vec3 iColor;
varying float vis;
varying float nodeType;

float pixelWidthRatio;

void main() {
  pixelWidthRatio = 1.0 / (vpSize.x * projectionMatrix[0][0]);
  oColor = outerColor;
  iColor = innerColor;

  mat4 mat = projectionMatrix * modelViewMatrix;
  vec4 originalPosition = mat * vec4(position, 1.0);
  float pixelWidth = originalPosition.w * pixelWidthRatio;
  float resRatio = 0.5;
  vec3 off = vec3(0.0,0.0,0.0);
  if (length(nDir) > 0.0) {
    off = normalize(nDir) * resRatio * (` + size + ` * 0.5 * pixelWidth + offset);
  }
  if (hovered > 0.0) {
    gl_PointSize = ` + size + ` * 1.2;
  } else {
    gl_PointSize = ` + size + `;
  }
  nodeType = type;
  vec3 offsetPt = position + off;
  vis = visible;
  gl_Position = mat * vec4(offsetPt, 1.0);
}
```

Fragment shader

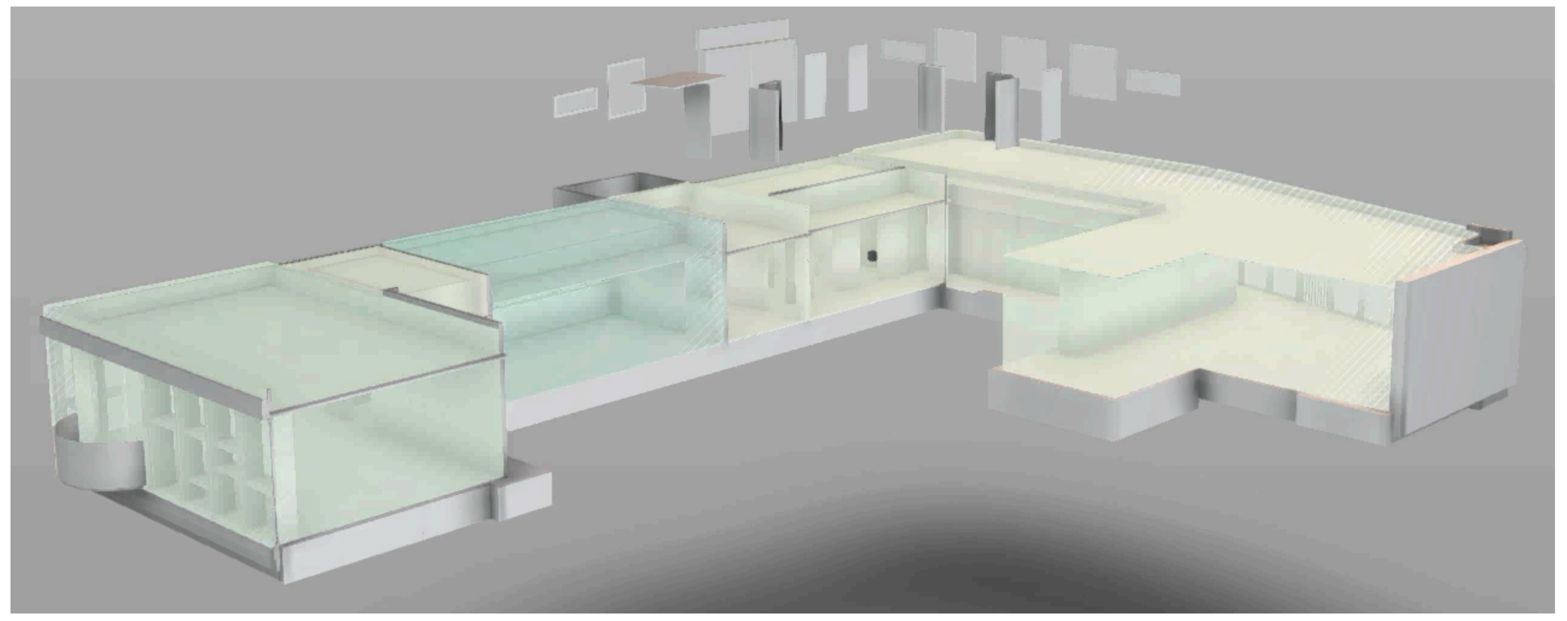
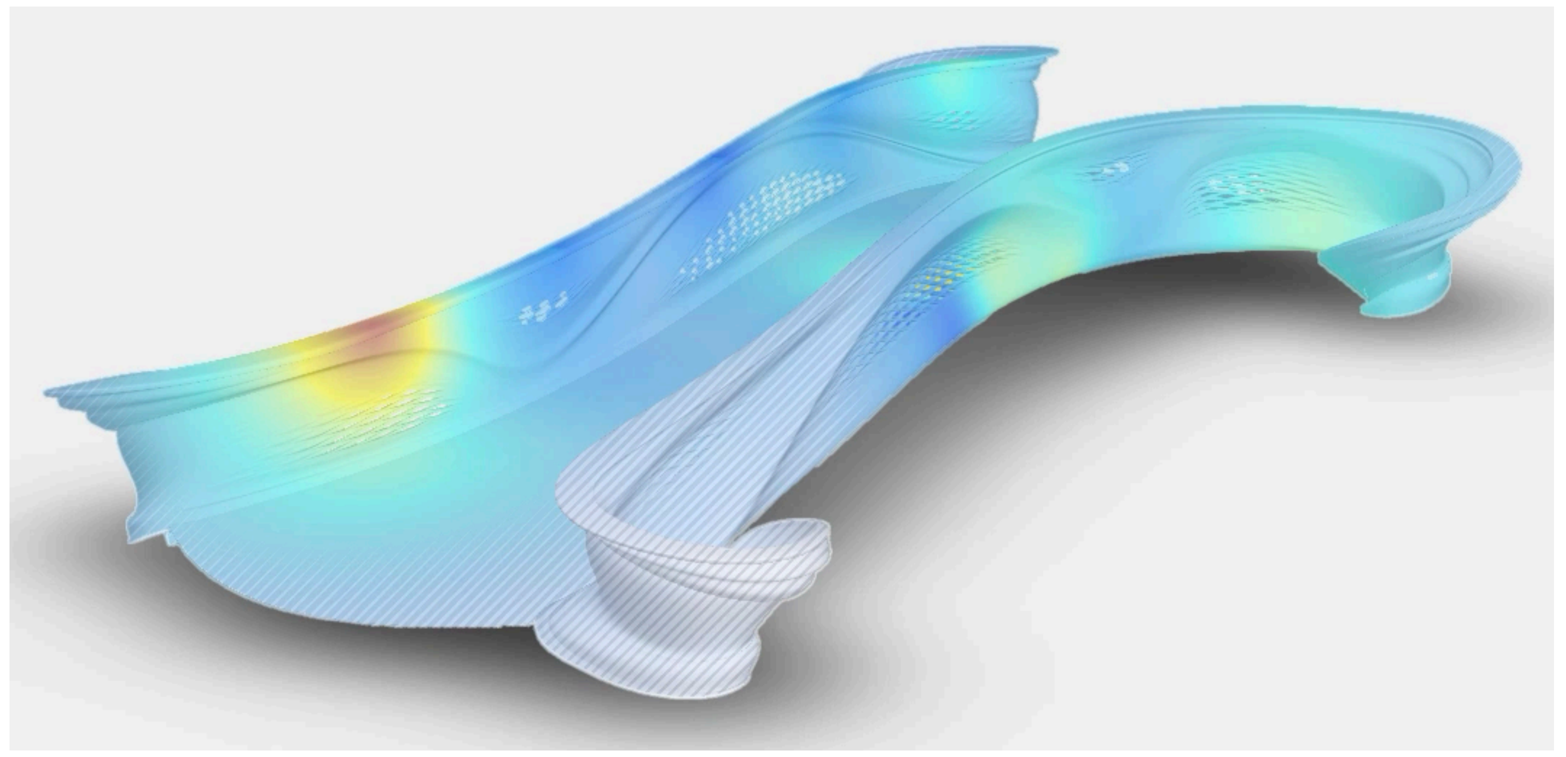
```
uniform sampler2D texture;
uniform sampler2D texture2;
uniform float groupID;
varying float vis;
varying float nodeType;
varying float vID;
void main() {
  if (vis < 0.5) discard;
  vec4 opacity;
  if (nodeType > 0.0) {
    opacity = texture2D(texture2, gl_PointCoord);
  } else {
    opacity = texture2D(texture, gl_PointCoord);
  }
  vec4 color = vec4(groupID, vID, 1.0, 1.0);
  if (opacity.a < 0.5) discard;
  gl_FragColor = color;
}
```

Custom Shaders – Sensor Hovering

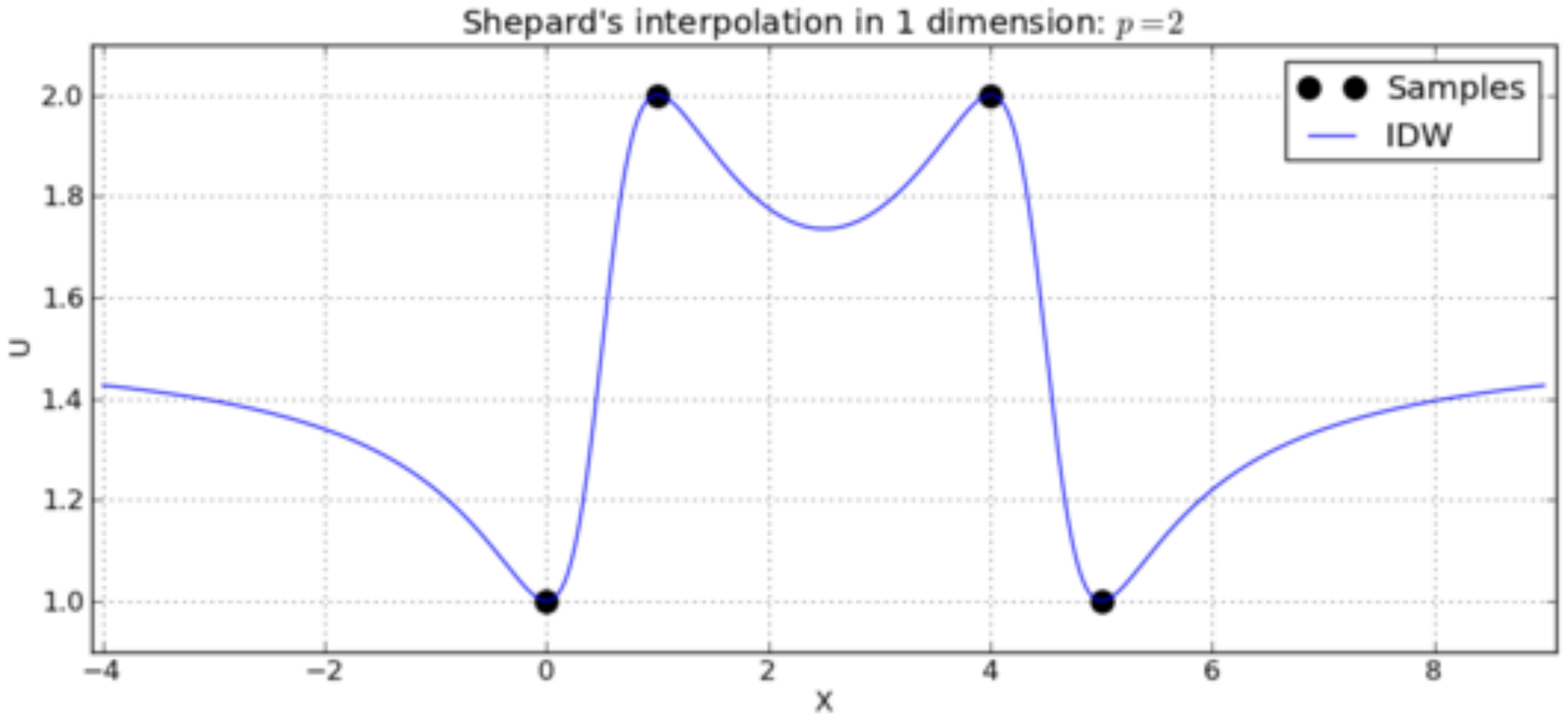


Custom Shaders – Heatmaps

- We use heatmaps to display large amounts of IoT data mapped to 3D
  - Mapped onto a 3D volume, such as a room
  - Mapped onto 3D geometry, such as a bridge
- Sensors are often located sparsely in these spaces
  - Need an approach to interpolate between the high-certainty locations
    - i.e. the sensors themselves
- We also animate them over time
  - A topic for another class

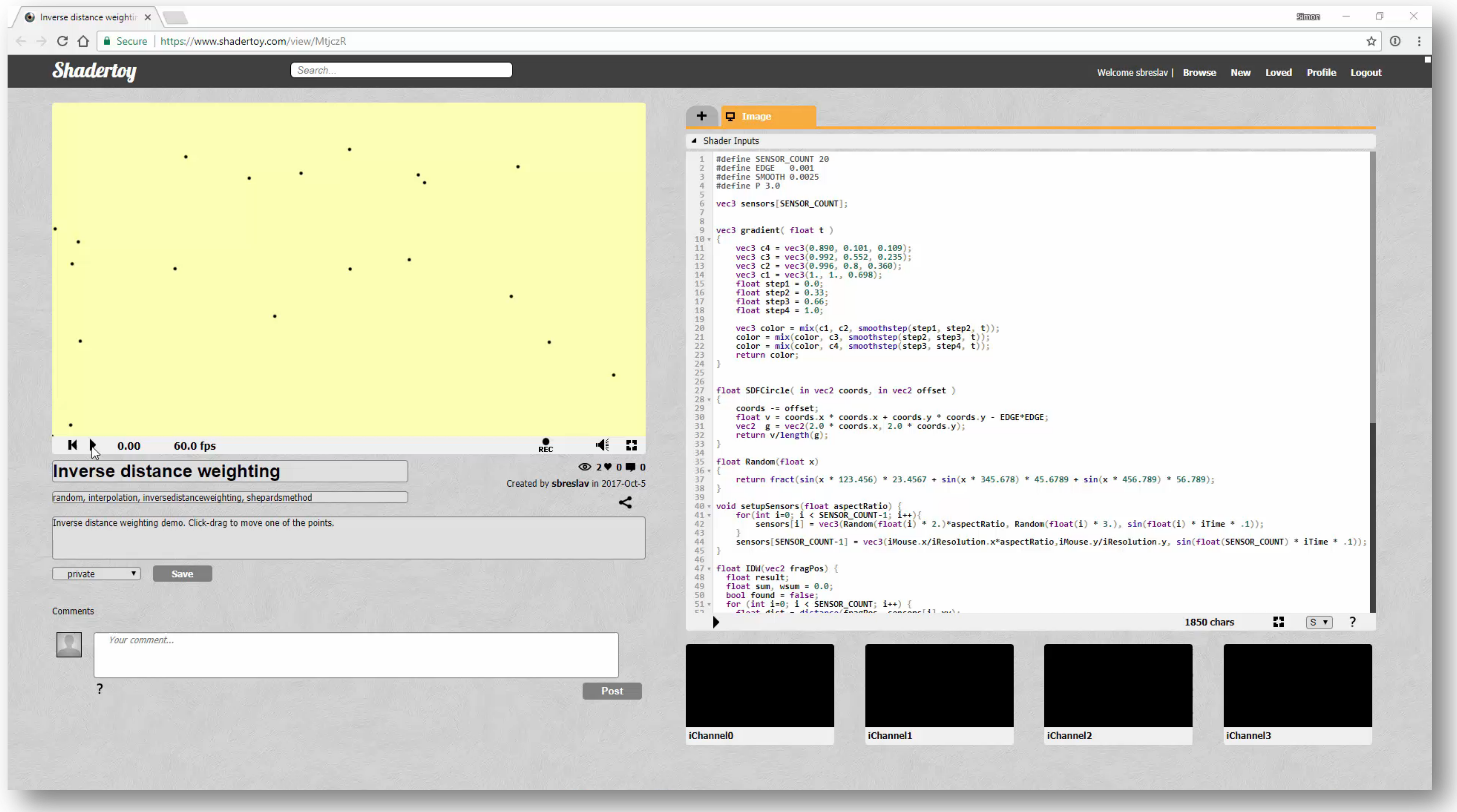
Custom Shaders – Heatmaps



Inverse Distance Weighting (a.k.a. Shepard's method)

Lots of other methods to interpolate values

Custom Shaders – Heatmaps


Inverse distance weighting
https://www.shadertoy.com/view/MtjczR
Shadertoy
Search...
Welcome sbreslav | Browse New Loved Profile Logout
Image
Shader Inputs
0.00
60.0 fps
REC
2 0 0
Created by sbreslav in 2017-Oct-5
private
Save
Comments
Your comment...
Post
1850 chars
iChannel0
iChannel1
iChannel2
iChannel3

Inverse distance weighting

random, interpolation, inversedistanceweighting, shepardsmethod

Inverse distance weighting demo. Click-drag to move one of the points.

```glsl
#define SENSOR_COUNT 20
#define EDGE   0.001
#define SMOOTH 0.0025
#define P 3.0

vec3 sensors[SENSOR_COUNT];


vec3 gradient( float t )
{
    vec3 c4 = vec3(0.890, 0.101, 0.109);
    vec3 c3 = vec3(0.992, 0.552, 0.235);
    vec3 c2 = vec3(0.996, 0.8, 0.360);
    vec3 c1 = vec3(1., 1., 0.698);
    float step1 = 0.0;
    float step2 = 0.33;
    float step3 = 0.66;
    float step4 = 1.0;

    vec3 color = mix(c1, c2, smoothstep(step1, step2, t));
    color = mix(color, c3, smoothstep(step2, step3, t));
    color = mix(color, c4, smoothstep(step3, step4, t));
    return color;
}


float SDFCircle( in vec2 coords, in vec2 offset )
{
    coords -= offset;
    float v = coords.x * coords.x + coords.y * coords.y - EDGE*EDGE;
    vec2  g = vec2(2.0 * coords.x, 2.0 * coords.y);
    return v/length(g);
}

float Random(float x)
{
    return fract(sin(x * 123.456) * 23.4567 + sin(x * 345.678) * 45.6789 + sin(x * 456.789) * 56.789);
}

void setupSensors(float aspectRatio) {
    for(int i=0; i < SENSOR_COUNT-1; i++){
        sensors[i] = vec3(Random(float(i) * 2.)*aspectRatio, Random(float(i) * 3.), sin(float(i) * iTime * .1));
    }
    sensors[SENSOR_COUNT-1] = vec3(iMouse.x/iResolution.x*aspectRatio,iMouse.y/iResolution.y, sin(float(SENSOR_COUNT) * iTime * .1));
}

float IDW(vec2 fragPos) {
  float result;
  float sum, wsum = 0.0;
  bool found = false;
  for (int i=0; i < SENSOR_COUNT; i++) {
```

[Give it a try online]()

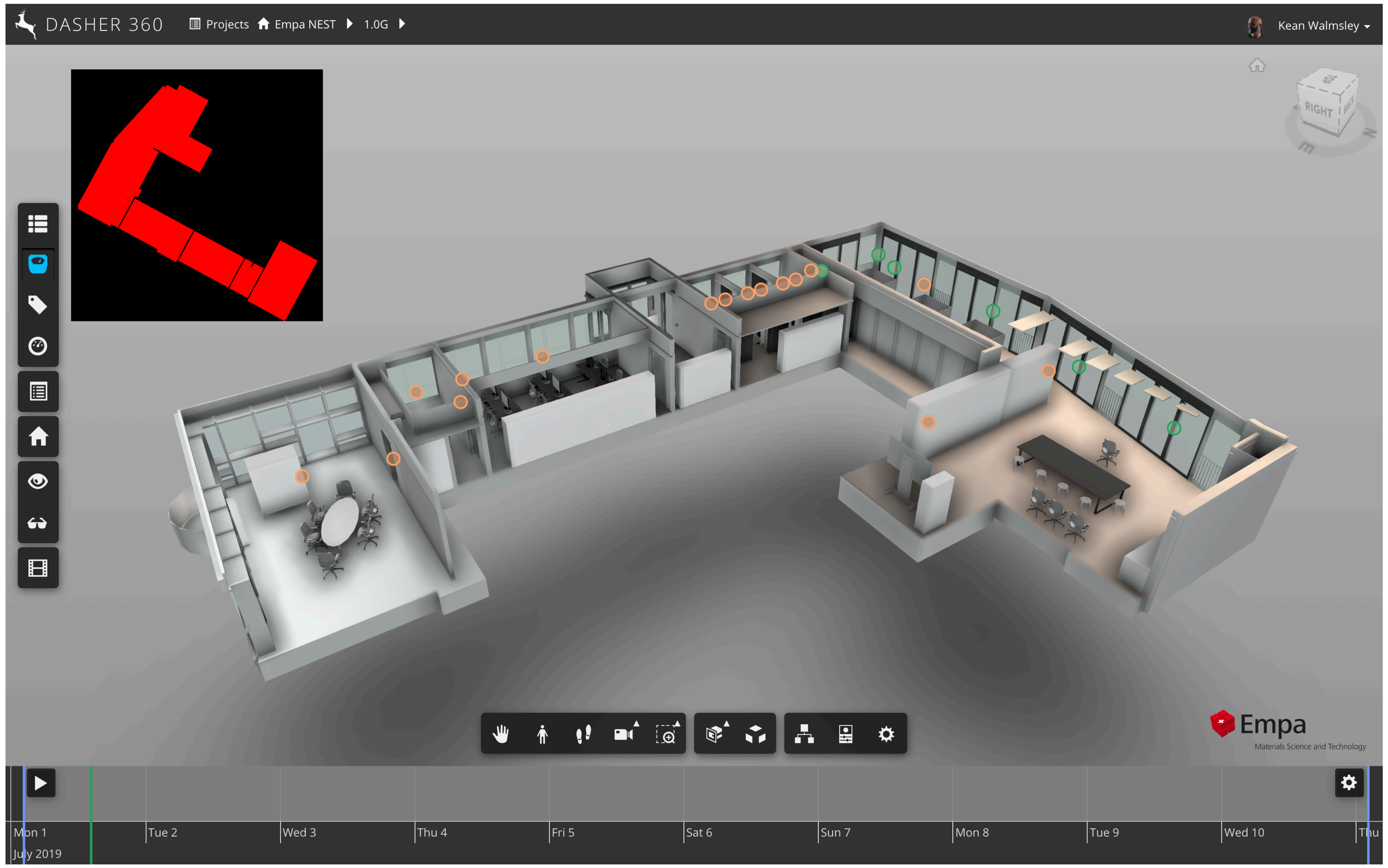
```glsl
…
float IDW(vec2 fragPos) {
  float sum, wsum = 0.0;
  for (int i=0; i < SENSOR_COUNT; i++) {
    float dist = distance(fragPos, sensors[i].xy);
    if( dist > 0.0) {
      float w = (1.0 / pow(dist, P));
      sum += (sensors[i].z * w);
      wsum += w;
    } else {
      return sensors[i].z;
    }
  }
  return sum / wsum;
}
…
```

GLSL

Custom Shaders – Room Identification

- We use another “reference shader” for rooms
  - One image per floor, with the room layout for that floor
- Various data gets encoded via the float texture
  - Room ID
  - Upper Z bound
  - Lower Z bound
- Allows us to identify the room containing a sensor or the FPV camera

Custom Shaders – Room Identification

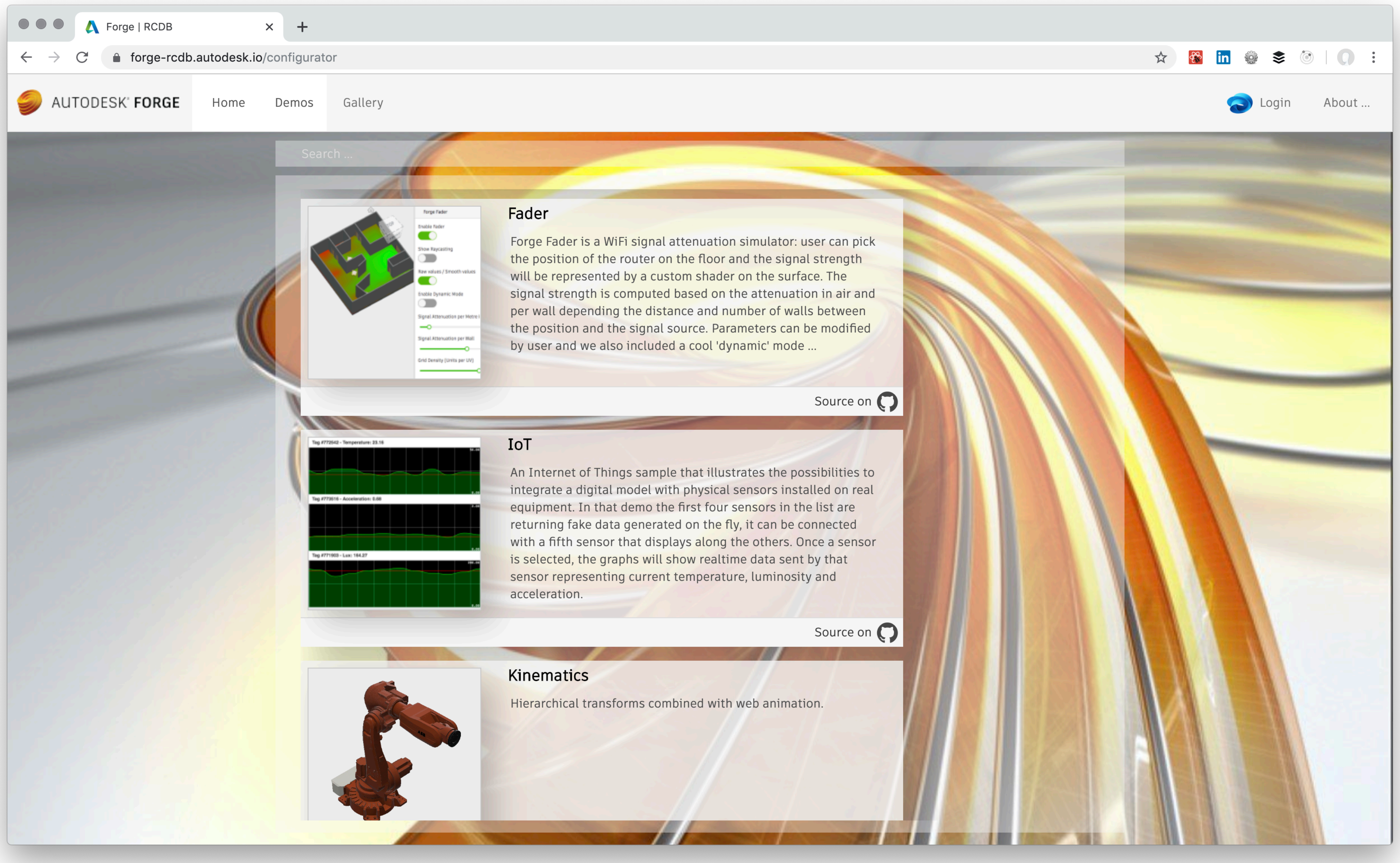


Shader Demo

- Sensor dots
- Sensor hovering
- Heatmaps
- Room identification

Useful Forge Examples

- Try online forge-rcdb.autodesk.io/configurator
- View source github.com/Autodesk-Forge/forge-rcdb.nodejs
  - Data Management
  - Extension Manager
  - Fader
  - IoT
  - Level Filter
  - Particle System
  - ScreenShot Manager



Fader

Forge Fader is a WiFi signal attenuation simulator: user can pick the position of the router on the floor and the signal strength will be represented by a custom shader on the surface. The signal strength is computed based on the attenuation in air and per wall depending the distance and number of walls between the position and the signal source. Parameters can be modified by user and we also included a cool 'dynamic' mode ...

Source on

IoT

An Internet of Things sample that illustrates the possibilities to integrate a digital model with physical sensors installed on real equipment. In that demo the first four sensors in the list are returning fake data generated on the fly, it can be connected with a fifth sensor that displays along the others. Once a sensor is selected, the graphs will show realtime data sent by that sensor representing current temperature, luminosity and acceleration.

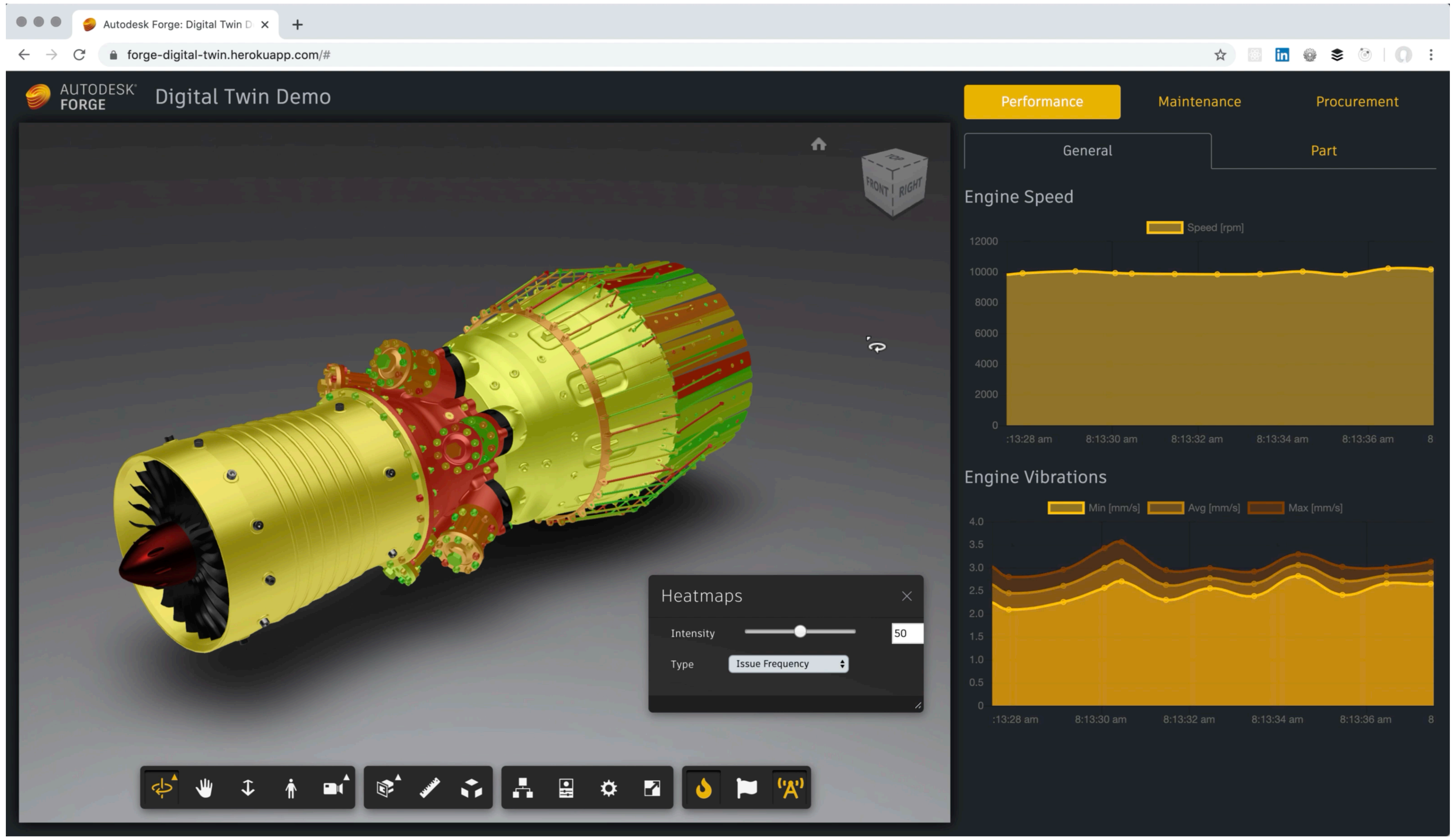
Source on

Kinematics

Hierarchical transforms combined with web animation.

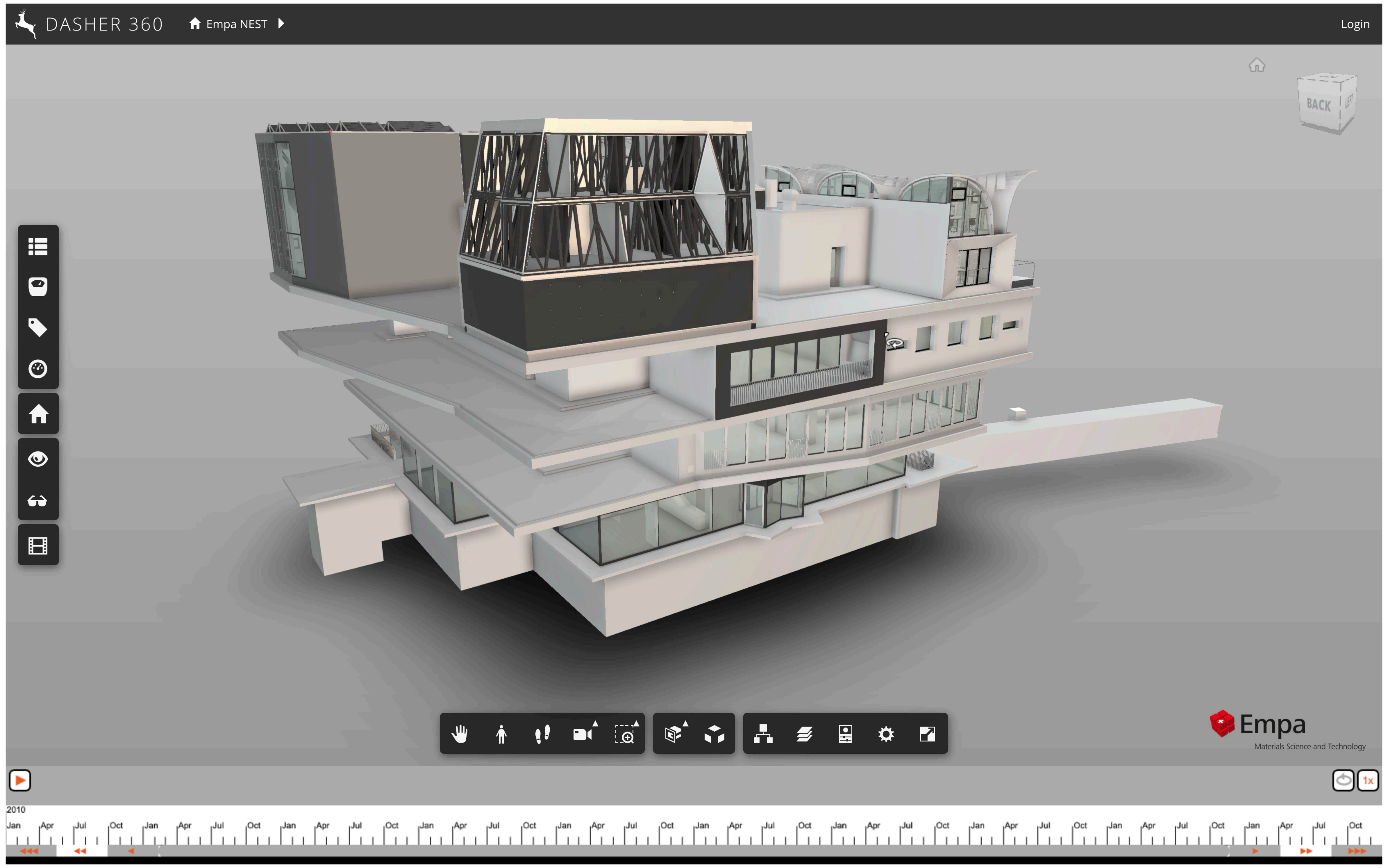
Useful Forge Examples

- Try online forge-digital-twin.herokuapp.com
- View source github.com/Autodesk-Forge/forge-digital-twin



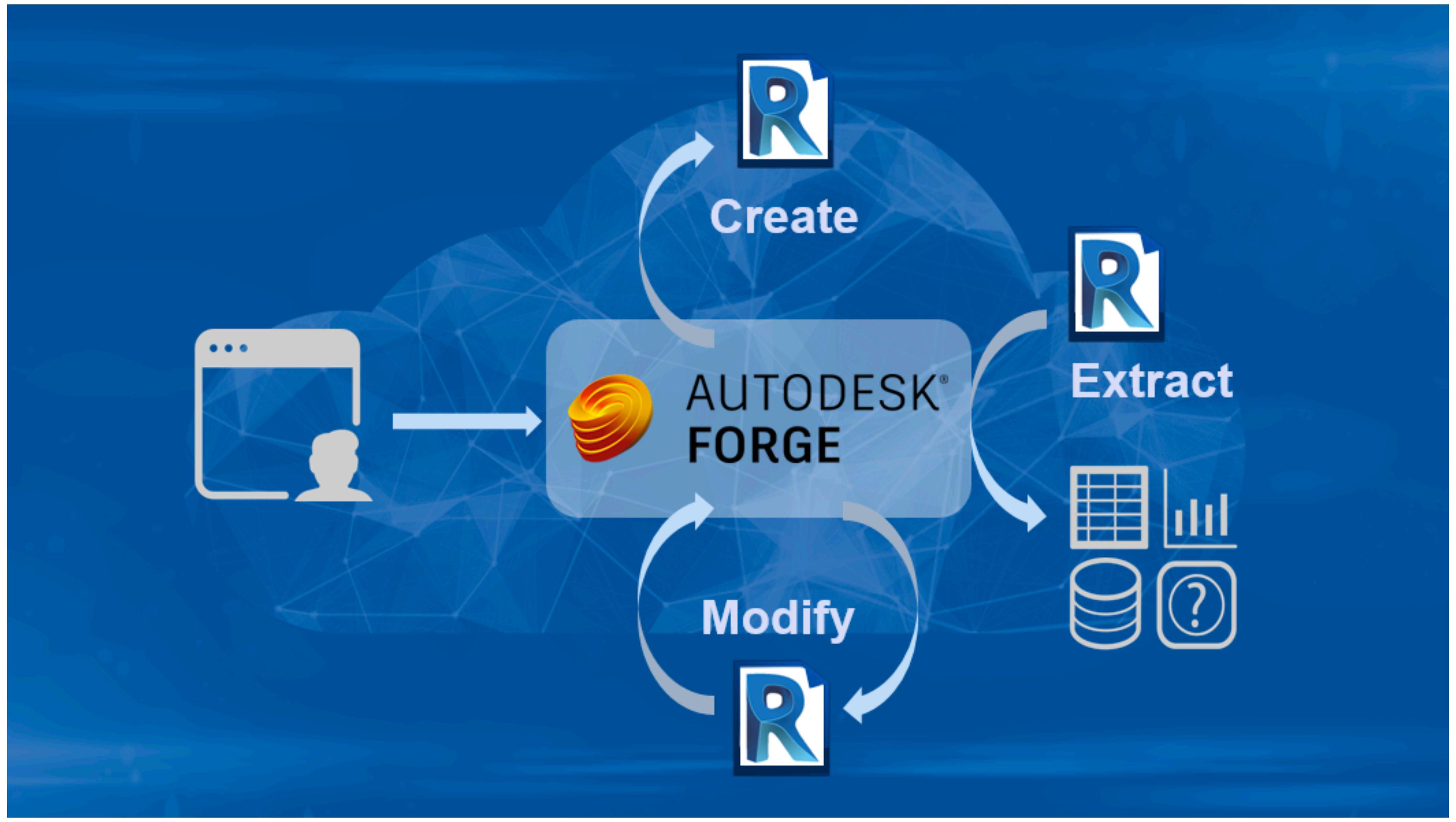
What’s Next For Dasher?

Future of Project Dasher



Future of Project Dasher

- Forge Design Automation for Revit is now available
  - Dasher 360 might update the source BIM
    - Sensors placed get added to the RVT
  - Also for model extraction
    - Avoid export to NWC to get rooms
    - Extract more building semantics



Future of Project Dasher

- We're actively exploring how this technology might be used by Owners/Operators
  - If interested in talking, please leave a business card or
  - Email project.oper8@autodesk.com
- Expect to hear more from us in the coming weeks/months
  - Join our mailing list via "Contact Us" on dasher360.com
  - Check Twitter (@keanw), LinkedIn (linkedin.com/in/keanw), or keanw.com

AUTODESK®

Make anything™