Paleontology in Python

Analyzing Dinosaur Trackways

Scott Ernst
Research Scientist
Paléojura (A16) Switzerland

Introduction

Data Science in Python

Applied to

Digital Paleontology

But First

Paleontology Background

History

The Jura Mountains
of Western Switzerland

History

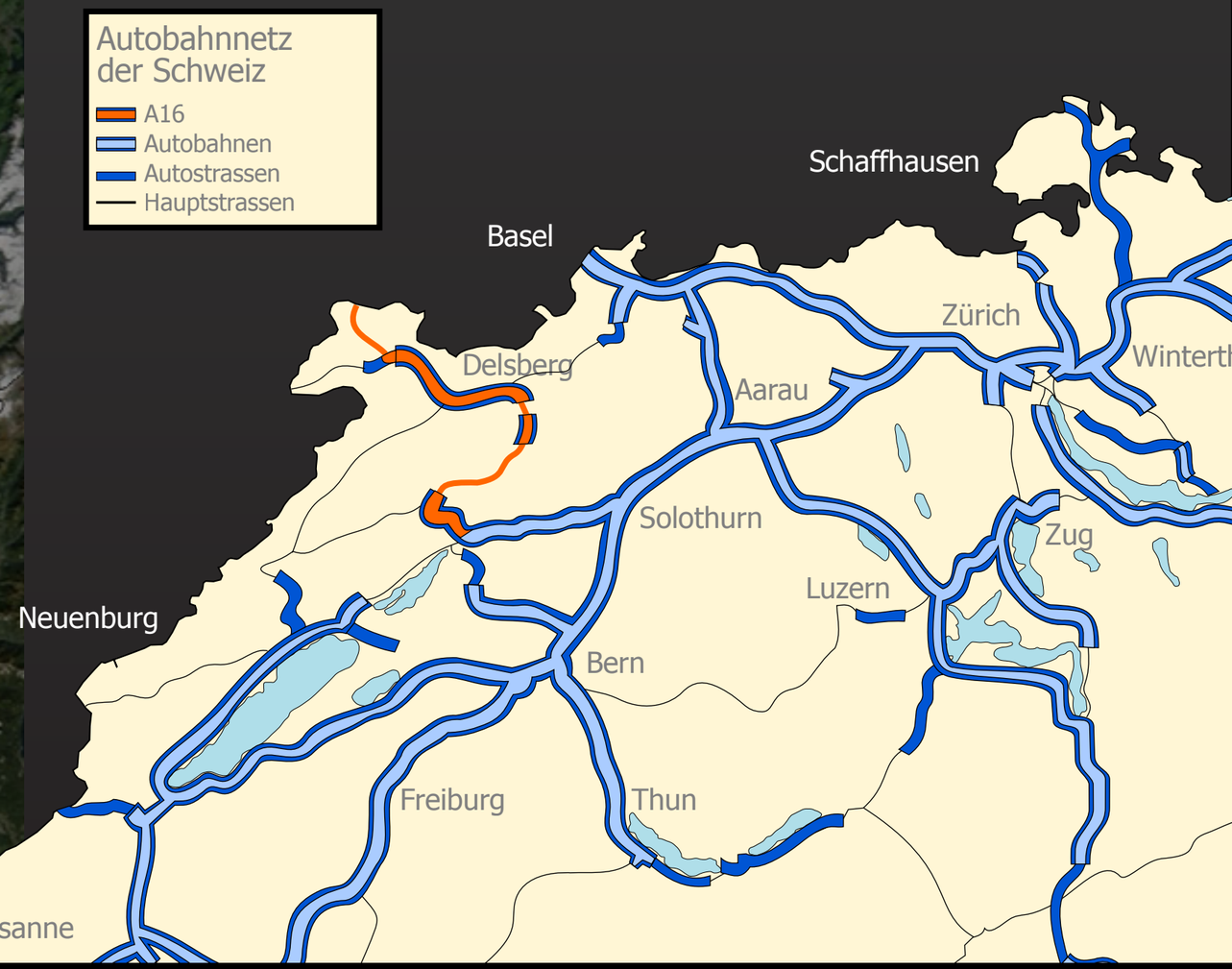
The Jura Mountains
of Western Switzerland

“Jura”ssic Period

History

About 15 Years Ago:

A16 Highway Construction



History

Construction Crews

Uncovered Dinosaur Tracks

Lots of Them

History

Swiss Government Funded:

A16/Paléojura Project

History

Years of Work
Excavating & Cataloging
Ahead of Road Construction

Results

COMMON

1-10

TRACKS

RARE

100+

TRACKS

EXTREMELY RARE

1,000+

TRACKS

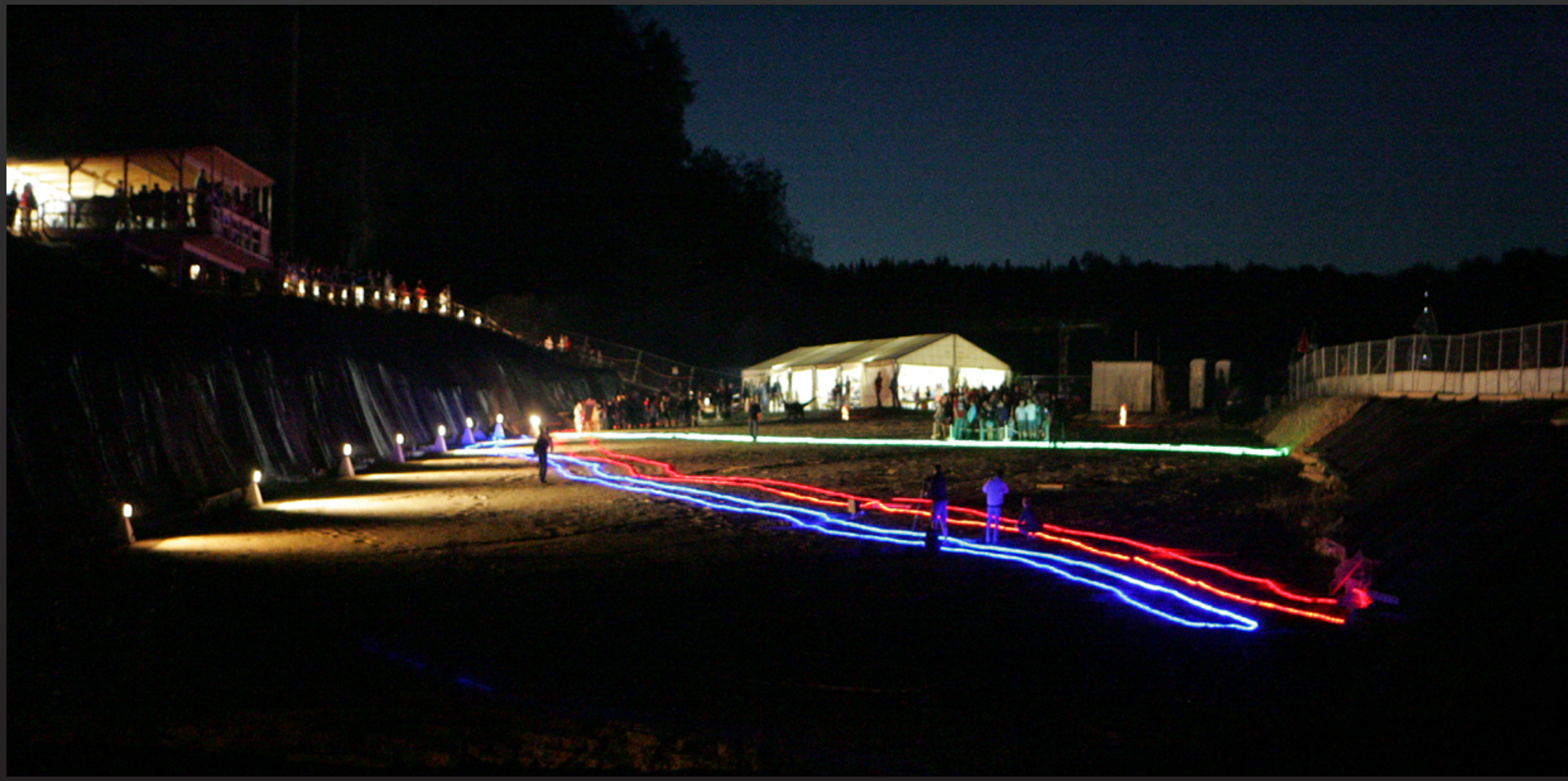
A16 PROJECT

10,000+

TRACKS

Results

Results

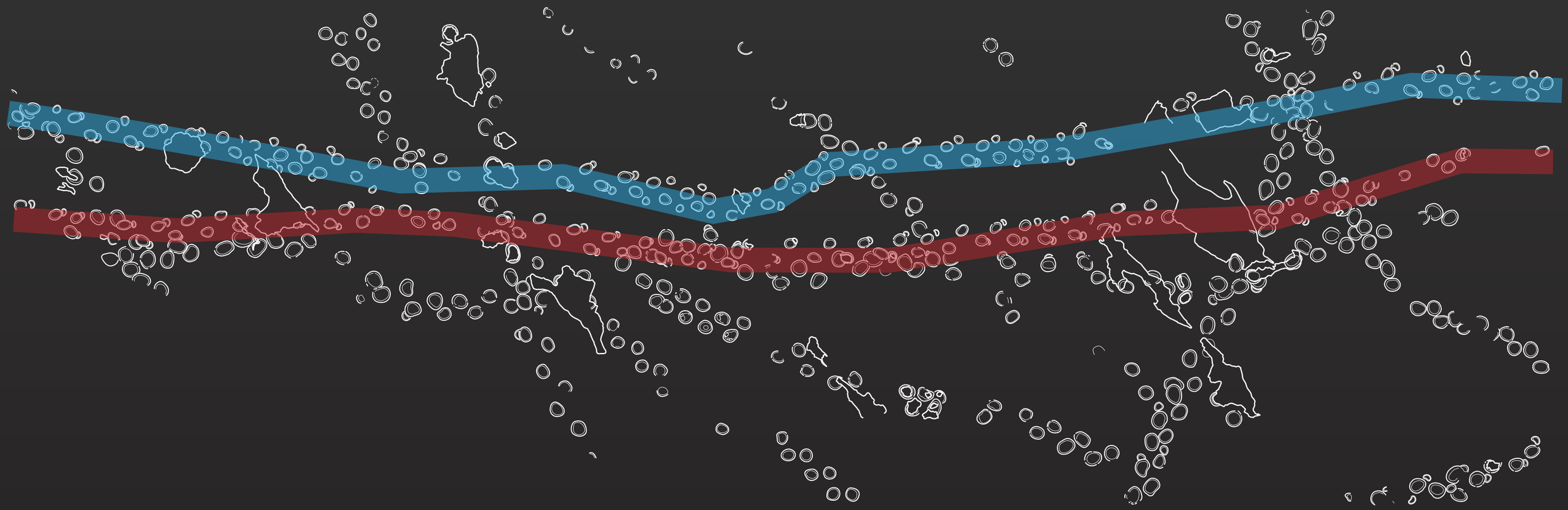


Results

Small Test Site

"Off the Highway"

Results



Were They Walking Together?

Results

Paleontology Excels at:

- Excavation
- Preservation
- Classification

Data
Modeling & Analysis

Results

Paléontologie A16 Organization

Daniel Marty
Research Director, Office de la culture

Wolfgang Hug
Head of the Paléontologie A16

Digital Paleontology

Kent Stevens
(Professor, University of Oregon)

Python + Maya

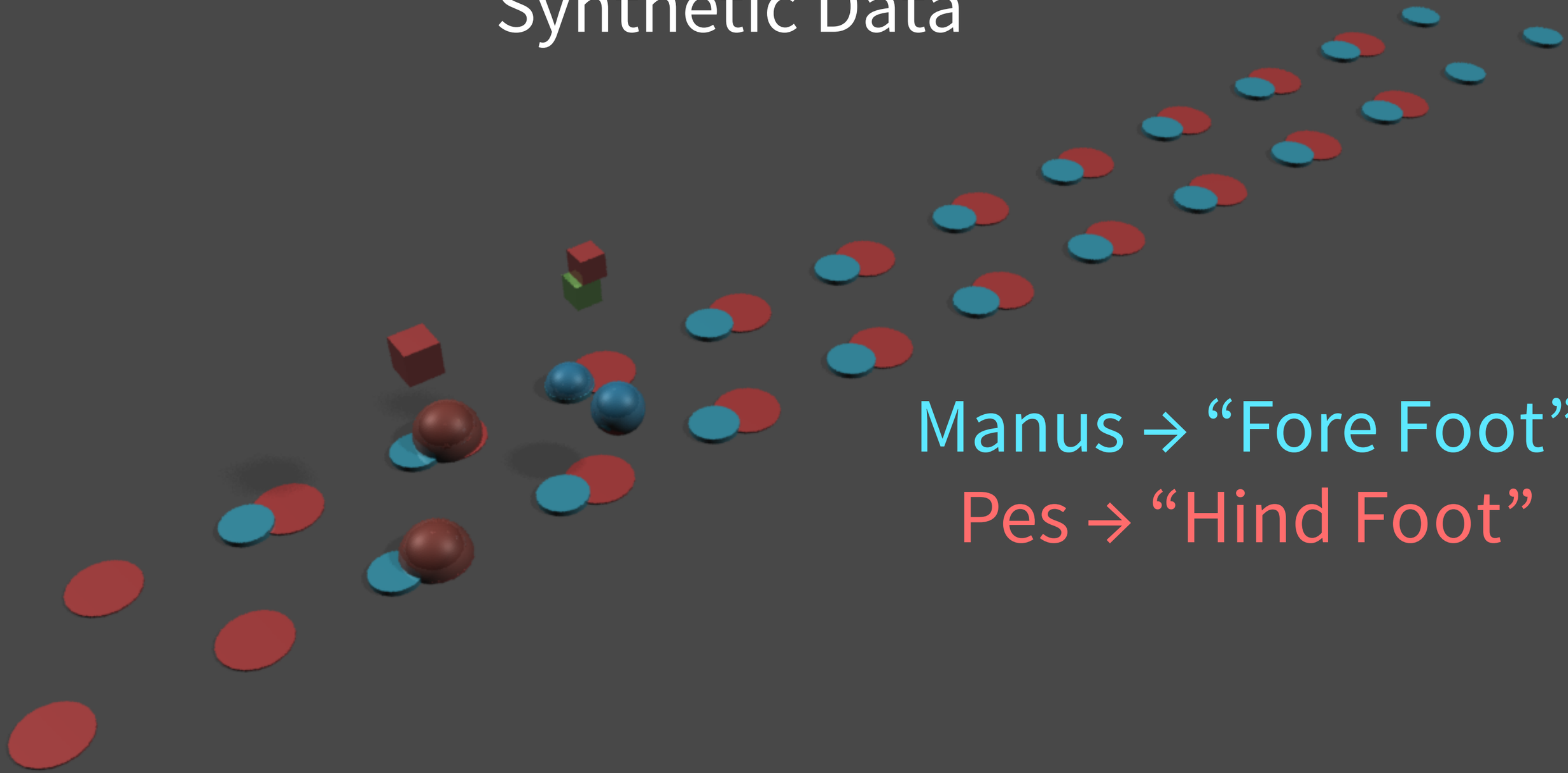
Double Negative
“Harry Potter”

Python + Maya

Lucasfilm
“Transformers”

Early Investigation

Synthetic Data

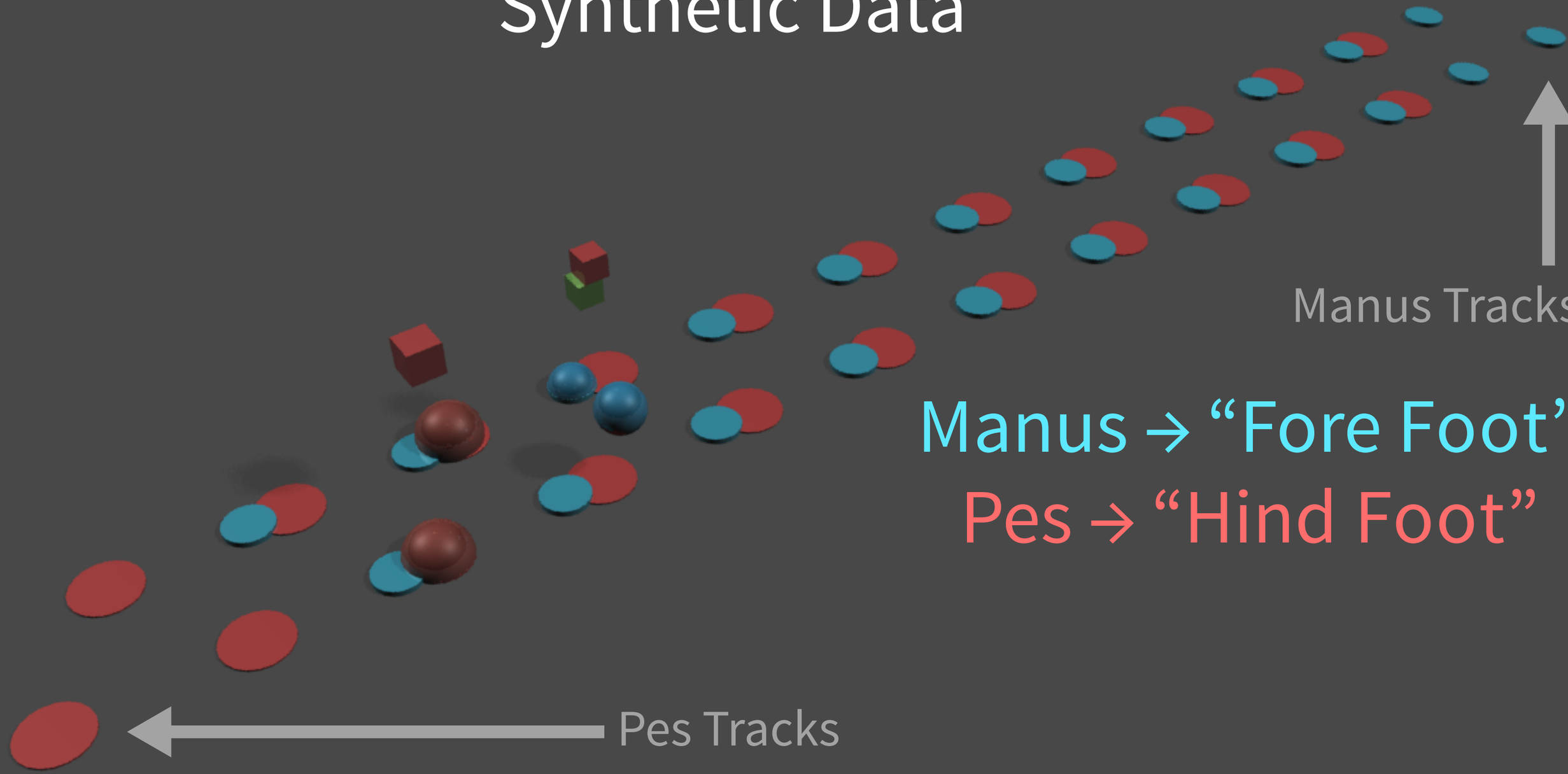


Manus → “Fore Foot”

Pes → “Hind Foot”

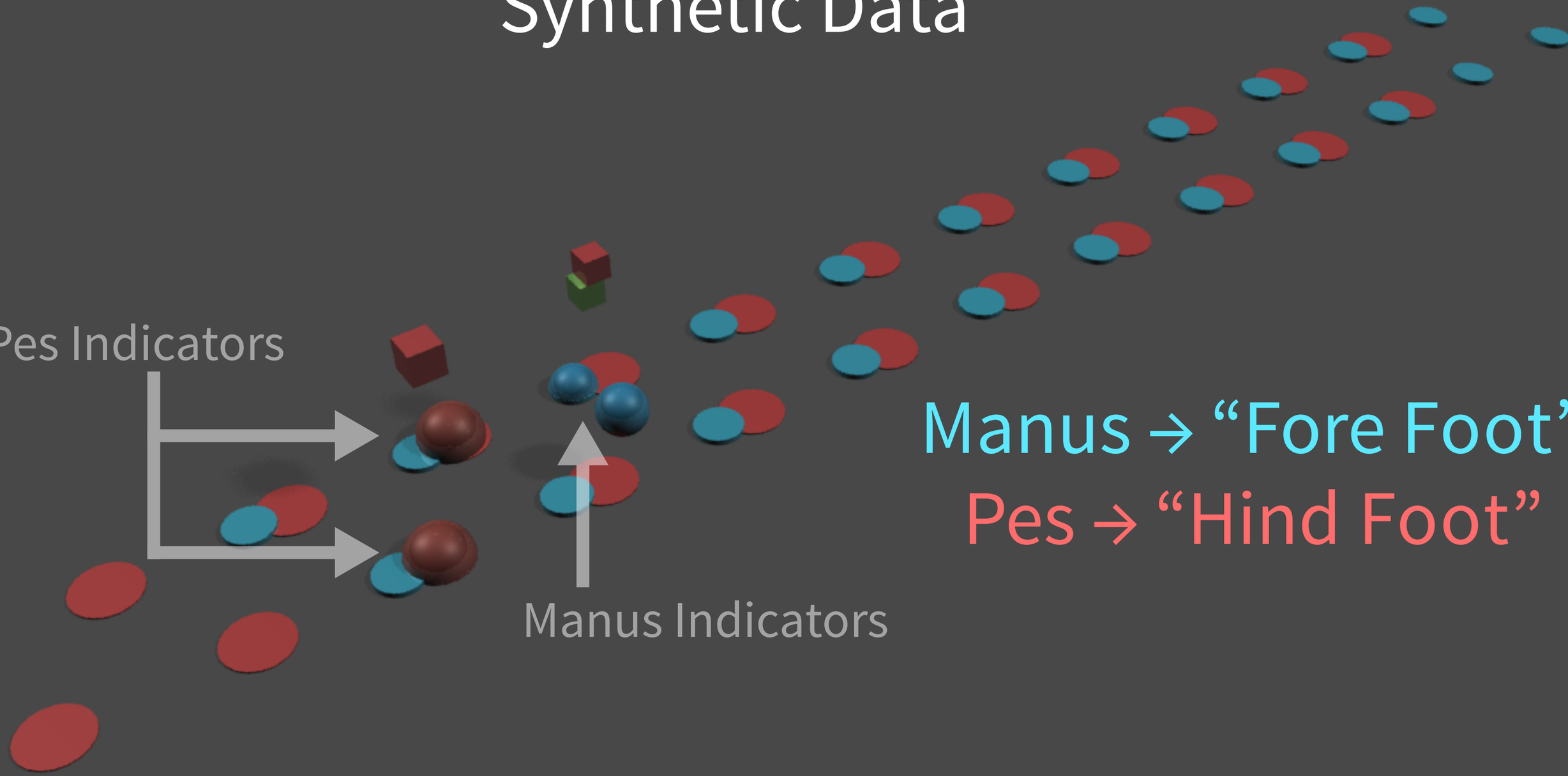
Early Investigation

Synthetic Data



Early Investigation

Synthetic Data



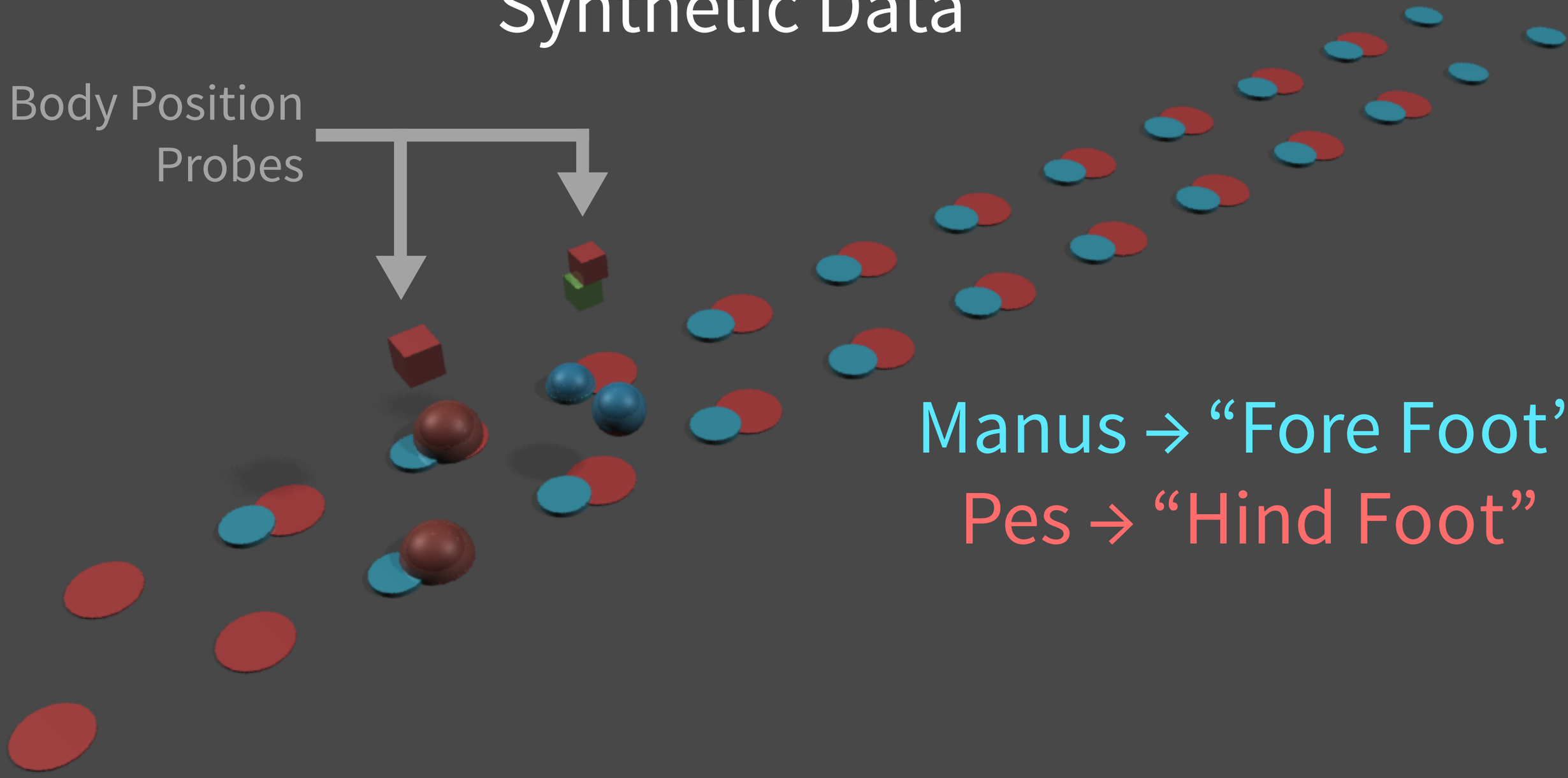
Manus → “Fore Foot”

Pes → “Hind Foot”

Early Investigation

Synthetic Data

Body Position Probes



Manus → “Fore Foot”

Pes → “Hind Foot”

Python & Maya Visualization

```
from maya import cmds
```

Python & Maya Visualization

```
34  from maya import cmds
35
36  # Create a sphere
37  sphereName, sphereShape = cmds.sphere()
38
39  # Move the created sphere to x=10
40  cmds.move(10, 0, 0, sphereName)
```

Python & Maya Visualization

```
34  from maya import cmds
35
36  # Create a sphere
37  sphereName, sphereShape = cmds.sphere()
38
39  # Move the created sphere to x=10
40  cmds.move(10, 0, 0, sphereName)
```

A

aaf2fcp	ambientLight	artAttrSkinPaintCtx	attrControlGrp
about	angleBetween	artAttrTool	attrEnumOptionMenu
addAttr	animCurveEditor	artBuildPaintMenu	attrEnumOptionMenuGrp
addDynamic	animDisplay	artFluidAttrCtx	attrFieldGrp
addExtension	animLayer	artPuttyCtx	attrFieldSliderGrp
addMetadata	animView	artSelectCtx	attributeInfo
addPP	annotate	artSetPaintCtx	attributeMenu
affectedNet	applyAttrPattern	artUserPaintCtx	attributeName
affects	applyMetadata	assembly	attributeQuery
aimConstraint	applyTake	assignCommand	attrNavigationControlGrp
air	arclen	assignInputDevice	audioTrack
aliasAttr	arcLenDimContext	assignViewportFactories	autoKeyframe
align	arcLengthDimension	attachCurve	autoPlace
alignCtx	arrayMapper	attachDeviceAttr	autoSave
alignCurve	art3dPaintCtx	attachSurface	
alignSurface	artAttrCtx	attrColorSliderGrp	
allNodeTypes	artAttrPaintVertexCtx	attrCompatibility	

B

bakeClip	bevelPlus	blendShape	boxZoomCtx
bakePartialHistory	bezierAnchorPreset	blendShapeEditor	bufferCurve
bakeResults	bezierAnchorState	blendShapePanel	buildBookmarkMenu
bakeSimulation	bezierCurveToNurbs	blendTwoAttr	buildKeyframeMenu
baseTemplate	bezierInfo	blindDataType	button
baseView	bindSkin	boneLattice	buttonManip
batchRender	binMembership	boundary	
bevel	blend2	boxDollyCtx	

C

cacheFile	clipSchedule	connectAttr	createEditor
cacheFileCombine	clipSchedulerOutliner	connectControl	createLayeredPsdFile
cacheFileMerge	closeCurve	connectDynamic	createNode
cacheFileTrack	closeSurface	connectionInfo	createRenderLayer
callbacks	cluster	connectJoint	createSubdivRegion
camera	cmdFileOutput	constrain	ctxAbort
cameraGet	cmdScrollFieldExecuter	constructionHistory	ctxCompletion
cameraView	cmdScrollFieldReporter	container	ctxEditMode
canCreateManip	cmdShell	containerBind	ctxTraverse
canvas	coarsenSubdivSelectionList	containerProxy	currentCtx
changeSubdivComponentDisplayLevel	collision	containerPublish	currentTime
changeSubdivRegion	color	containerTemplate	currentTimeCtx
channelBox	colorAtPoint	containerView	currentUnit
character	colorEditor	contextInfo	curve
characterize	colorIndex	control	curveAddPtCtx
characterMap	colorIndexSliderGrp	convertTiffToPsd	curveCVCtx
checkBox	colorSliderButtonGrp	convertLightmap	curveEditorCtx
checkBoxGrp	colorSliderGrp	convertSolidTx	curveEPCtx
checkDefaultRenderGlobals	columnLayout	convertTessellation	curveIntersect
choice	commandEcho	convertUnit	curveMoveEPCtx
circle	commandLine	copyAttr	curveOnSurface
circularFillet	commandPort	copyDeformerWeights	curveRGBColor
clearCache	componentBox	copyFlexor	curveSketchCtx
clip	componentEditor	copyKey	cutKey
clipEditor	condition	copySkinWeights	cycleCheck
clipEditorCurrentTimeCtx	cone	createAttrPatterns	cylinder
clipMatching	confirmDialog	createDisplayLayer	

D

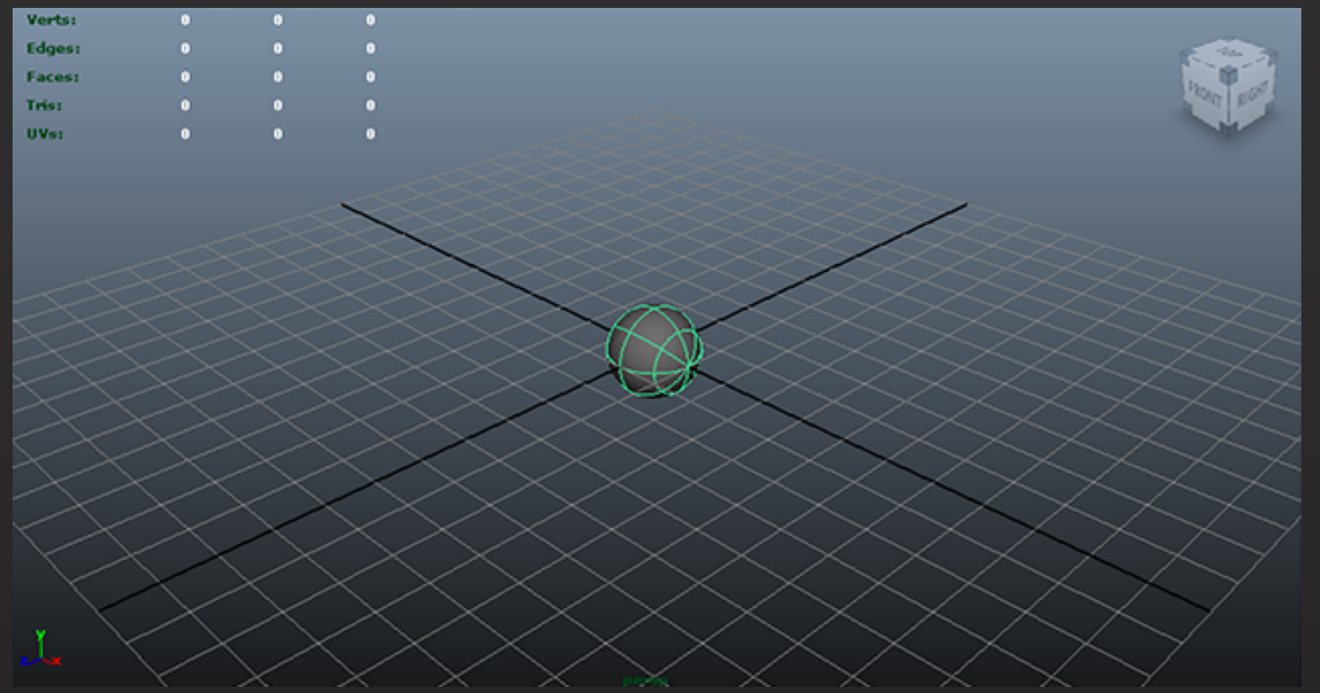
dagObjectCompare	detachDeviceAttr	displayAffected	dragAttrContext
dagPose	detachSurface	displayColor	draggerContext
dataStructure	deviceEditor	displayCull	dropoffLocator
date	deviceManager	displayLevelOfDetail	duplicate
dbcount	devicePanel	displayPref	duplicateCurve
dbmessage	dgdirty	displayRGBColor	duplicateSurface
dbpeek	dgeval	displaySmoothness	dynamicLoad
defaultLightListCheckBox	dgInfo	displayStats	dynCache
defaultNavigation	dgmodified	displayString	dynControl
defineDataServer	dgtimer	displaySurface	dynExport
defineVirtualDevice	dimWhen	distanceDimContext	dynExpression
deformer	directionalLight	distanceDimension	dynGlobals
deformerWeights	directKeyCtx	doBlur	dynPaintEditor
delete	dirmap	dockControl	dynParticleCtx
deleteAttr	disable	dolly	dynPref
deleteAttrPattern	disconnectAttr	dollyCtx	
deleteExtension	disconnectJoint	dopeSheetEditor	
deleteUI	diskCache	doubleProfileBirailSurface	
detachCurve	displacementToPoly	drag	

E

editDisplayLayerGlobals	editRenderLayerMembers	eval	expressionEditorListen

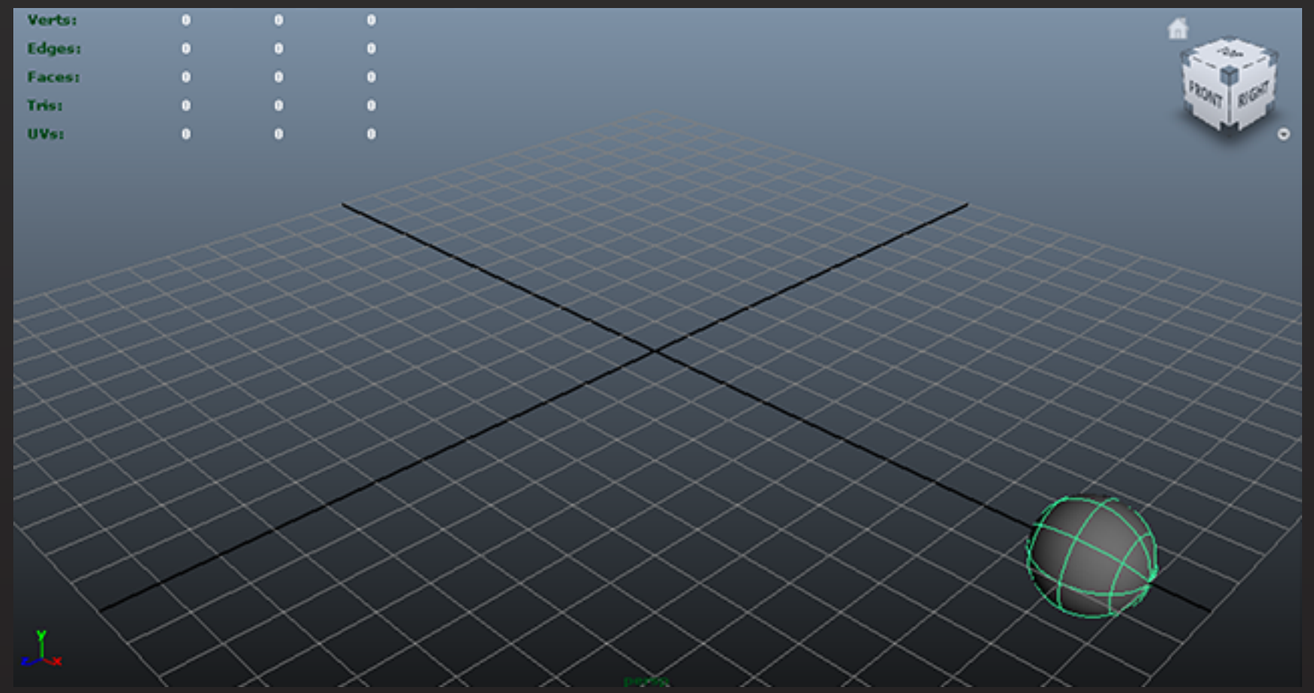
Python & Maya Visualization

```
34  from maya import cmds
35
36  # Create a sphere
37  sphereName, sphereShape = cmds.sphere()
38
39  # Move the created sphere to x=10
40  cmds.move(10, 0, 0, sphereName)
```



Python & Maya Visualization

```
34  from maya import cmds
35
36  # Create a sphere
37  sphereName, sphereShape = cmds.sphere()
38
39  # Move the created sphere to x=10
40  cmds.move(10, 0, 0, sphereName)
```



Python & Maya Visualization

```
 6  def createTrackwayVisualization(trackway):
 7
 8      references = []
 9
10      for foot in trackway:
11          references.extend(createMarchingSphere(foot))
12
13      references.append(createHipsIndicator(trackway))
14      references.append(createShoulderIndicator(trackway))
15
16      group = cmds.group(*references, world=True, name=name)
17      cmds.select(group)
18
19      applyShaders(trackway)
20      createRenderEnvironment()
21
22      minTime = min(0, int(cmds.playbackOptions(query=True, minTime=True)
23      maxTime = trackway.duration
24
25      cmds.playbackOptions(
26          minTime=minTime, animationStartTime=minTime,
27          maxTime= maxTime, animationEndTime=maxTime)
28
29      cmds.currentTime(0, update=True)
30      cmds.select(group)
```

Python & Maya Visualization

```
6   def createTrackwayVisualization(trackway):
7
8       references = []
9
10      for foot in trackway:
11          references.extend(createMarchingSphere
12
13      references.append(createHipsIndicator(
14      references.append(createShoulderIndicator(
15
16      group = cmds.group(*references, world=Tru
17      cmds.select(group)
18
19      applyShaders(trackway)
20      createRenderEnvironment()
21
22      minTime = min(0, int(cmds.playbackOptions(
23      maxTime = trackway.duration
24
25      cmds.playbackOptions(
26          minTime=minTime, animationStartTime=minTime,
27          maxTime= maxTime, animationEndTime=maxTime)
28
29      cmds.currentTime(0, update=True)
30      cmds.select(group)
```

Synthetic Data Object

Python & Maya Visualization

1

```
6  def createTrackwayVisualization(trackway):
7
8      references = []
9
10     for foot in trackway:
11         references.extend(createMarchingSphere(foot))
12
13     references.append(createHipsIndicator(trackway))
14     references.append(createShoulderIndicator(trackway))
15
16     group = cmds.group(*references, world=True, name=name)
17     cmds.select(group)
18
19     applyShaders(trackway)
20     createRenderEnvironment()
21
22     minTime = min(0, int(cmds.playbackOptions(query=True, minTime=True)
23     maxTime = trackway.duration
24
25     cmds.playbackOptions(
26         minTime=minTime, animationStartTime=minTime,
27         maxTime= maxTime, animationEndTime=maxTime)
28
29     cmds.currentTime(0, update=True)
30     cmds.select(group)
```

Python & Maya Visualization

```
6  def createTrackwayVisualization(trackway):
7
8      references = []
9
10     for foot in trackway:
11         references.extend(createMarchingSphere(foot))
12
13     references.append(createHipsIndicator(trackway))
14     references.append(createShoulderIndicator(trackway))
15
16     group = cmds.group(*references, world=True, name=name)
17     cmds.select(group)
18
19     applyShaders(trackway)
20     createRenderEnvironment()
21
22     minTime = min(0, int(cmds.playbackOptions(query=True, minTime=True)
23     maxTime = trackway.duration
24
25     cmds.playbackOptions(
26         minTime=minTime, animationStartTime=minTime,
27         maxTime= maxTime, animationEndTime=maxTime)
28
29     cmds.currentTime(0, update=True)
30     cmds.select(group)
```

1

2

Python & Maya Visualization

```
 6  def createTrackwayVisualization(trackway):
 7
 8      references = []
 9
10      for foot in trackway:
11          references.extend(createMarchingSphere(foot))
12
13      references.append(createHipsIndicator(trackway))
14      references.append(createShoulderIndicator(trackway))
15
16      group = cmds.group(*references, world=True, name=name)
17      cmds.select(group)
18
19      applyShaders(trackway)
20      createRenderEnvironment()
21
22      minTime = min(0, int(cmds.playbackOptions(query=True, minTime=True)
23      maxTime = trackway.duration
24
25      cmds.playbackOptions(
26          minTime=minTime, animationStartTime=minTime,
27          maxTime= maxTime, animationEndTime=maxTime)
28
29      cmds.currentTime(0, update=True)
30      cmds.select(group)
```

1

2

3

Python & Maya Visualization

```
6  def createTrackwayVisualization(trackway):
7
8      references = []
9
10     for foot in trackway:
11         references.extend(createMarchingSphere(foot))
12
13     references.append(createHipsIndicator(trackway))
14     references.append(createShoulderIndicator(trackway))
15
16     group = cmds.group(*references, world=True, name=name)
17     cmds.select(group)
18
19     applyShaders(trackway)
20     createRenderEnvironment()
21
22     minTime = min(0, int(cmds.playbackOptions(query=True, minTime=True)
23     maxTime = trackway.duration
24
25     cmds.playbackOptions(
26         minTime=minTime, animationStartTime=minTime,
27         maxTime= maxTime, animationEndTime=maxTime)
28
29     cmds.currentTime(0, update=True)
30     cmds.select(group)
```

1

2

3

4

Python & Maya Visualization

```
32  def createMarchingSphere(foot):
33      sphereName, sphereShape = cmds.polySphere(radius=foot.radius, name=foot.name)
34      reference = [sphereName]
35
36      for keyFrame in foot.keys:
37          for keyable in keyFrame.attributes:
38              cmds.setKeyframe(
39                  sphereName,
40                  attribute=keyable.name,
41                  time=keyFrame.time,
42                  value=keyable.value)
43
44          if keyFrame.isLandEvent:
45              trackName, trackShape = cmds.polyCylinder(
46                  name=foot.label + '_print_1', # Maya numbers automatically
47                  radius=foot.radius,
48                  height=1.0 if foot.isPes else 5.0)
49
50              cmds.move(keyFrame.value.x, keyFrame.value.y, keyFrame.value.z, trackName)
51              reference.append(trackName)
52
53      return reference
```

Python & Maya Visualization

```
32  def createMarchingSphere(foot):
33      sphereName, sphereShape = cmds.polySphere(radius=foot.radius, name=foot.name)
34      reference = [sphereName]
35
36      for keyFrame in foot.keys:
37          for keyable in keyFrame.attributes:
38              cmds.setKeyframe(
39                  sphereName,
40                  attribute=keyable.name,
41                  time=keyFrame.time,
42                  value=keyable.value)
43
44          if keyFrame.isLandEvent:
45              trackName, trackShape = cmds.polyCylinder(
46                  name=foot.label + '_print_1', # Maya numbers automatically
47                  radius=foot.radius,
48                  height=1.0 if foot.isPes else 5.0)
49
50              cmds.move(keyFrame.value.x, keyFrame.value.y, keyFrame.value.z, trackName)
51              reference.append(trackName)
52
53      return reference
```

Create Sphere Indicator

Python & Maya Visualization

```
32  def createMarchingSphere(foot):
33      sphereName, sphereShape = cmds.polySphere(radius=foot.radius, name=foot.name)
34      reference = [sphereName]
35
36      for keyFrame in foot.keys:
37          for keyable in keyFrame.attributes:
38              cmds.setKeyframe(
39                  sphereName,
40                  attribute=keyable.name,
41                  time=keyFrame.time,
42                  value=keyable.value)
43
44          if keyFrame.isLandEvent:
45              trackName, trackShape = cmds.polyCylinder(
46                  name=foot.label + '_print_1', # Maya numbers automatically
47                  radius=foot.radius,
48                  height=1.0 if foot.isPes else 5.0)
49
50              cmds.move(keyFrame.value.x, keyFrame.value.y, keyFrame.value.z, trackName)
51              reference.append(trackName)
52
53      return reference
```

Set Key Frames

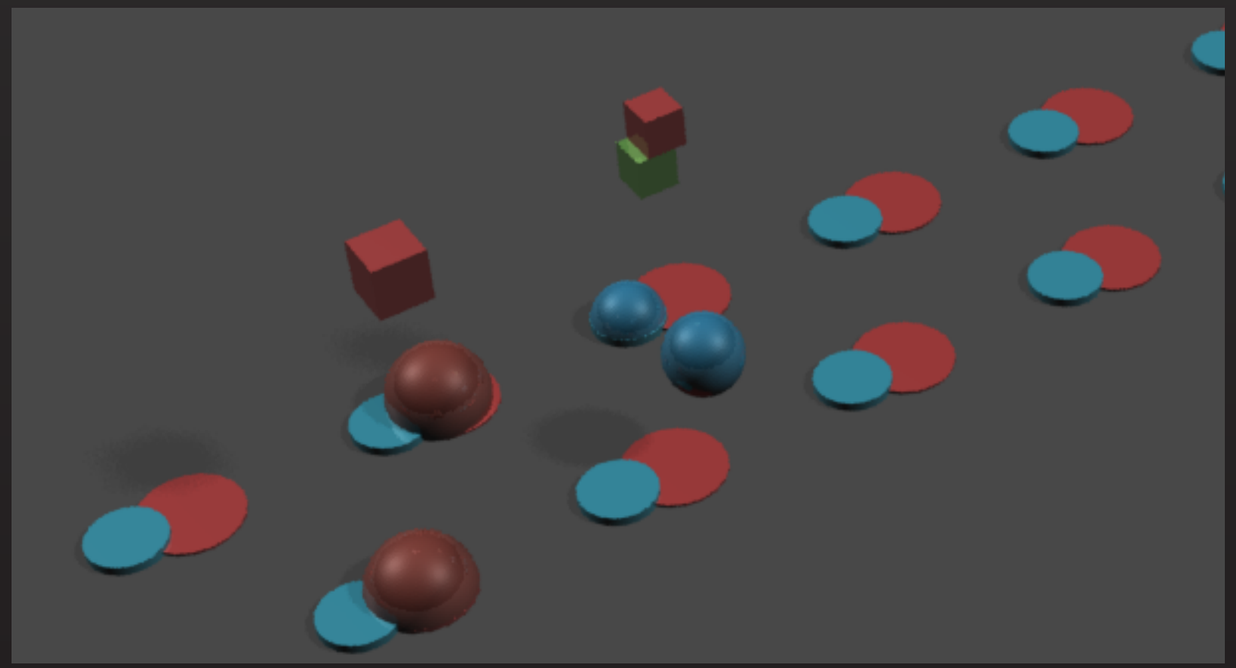
Python & Maya Visualization

```
32  def createMarchingSphere(foot):
33      sphereName, sphereShape = cmds.polySphere(radius=foot.radius, name=foot.name)
34      reference = [sphereName]
35
36      for keyFrame in foot.keys:
37          for keyable in keyFrame.attributes:
38              cmds.setKeyframe(
39                  sphereName,
40                  attribute=keyable.name,
41                  time=keyFrame.time,
42                  value=keyable.value)
43
44          if keyFrame.isLandEvent:
45              trackName, trackShape = cmds.polyCylinder(
46                  name=foot.label + '_print_1', # Maya numbers automatically
47                  radius=foot.radius,
48                  height=1.0 if foot.isPes else 5.0)
49
50              cmds.move(keyFrame.value.x, keyFrame.value.y, keyFrame.value.z, trackName)
51              reference.append(trackName)
52
53      return reference
```

Create Foot Print

Python & Maya Visualization

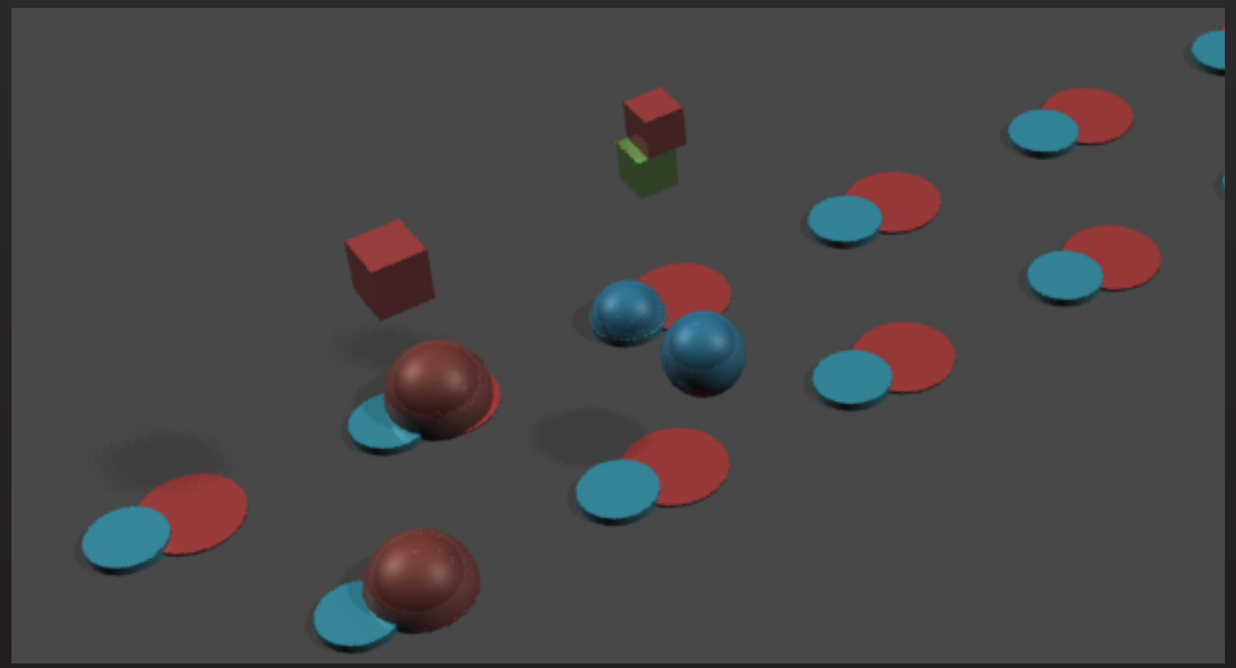
```
55  def createHipsIndicator(trackway):
56      hips, shape = cmds.polyCube(name='hips', width=20, height=20, depth=20)
57      cmds.move(0, 100, 0, hips)
58      references.append(hips)
59
60      shoulders, shape = cmds.polyCube(name='shoulders_locked', width=15, height=15, depth=15)
61      cmds.move(0, 115, trackway.spineLength, shoulders)
62      cmds.parent(shoulders, hips, absolute=True)
63
64      pes = trackway.pes
65      cmds.expression("%s.z = 0.5*abs(%s.z - %s.z) + min(%s.z, %s.z)" % (
66          hips, pes.left, pes.right, pes.left, pes.right))
67
68      return hips
```



Python & Maya Visualization

Hips Indicator

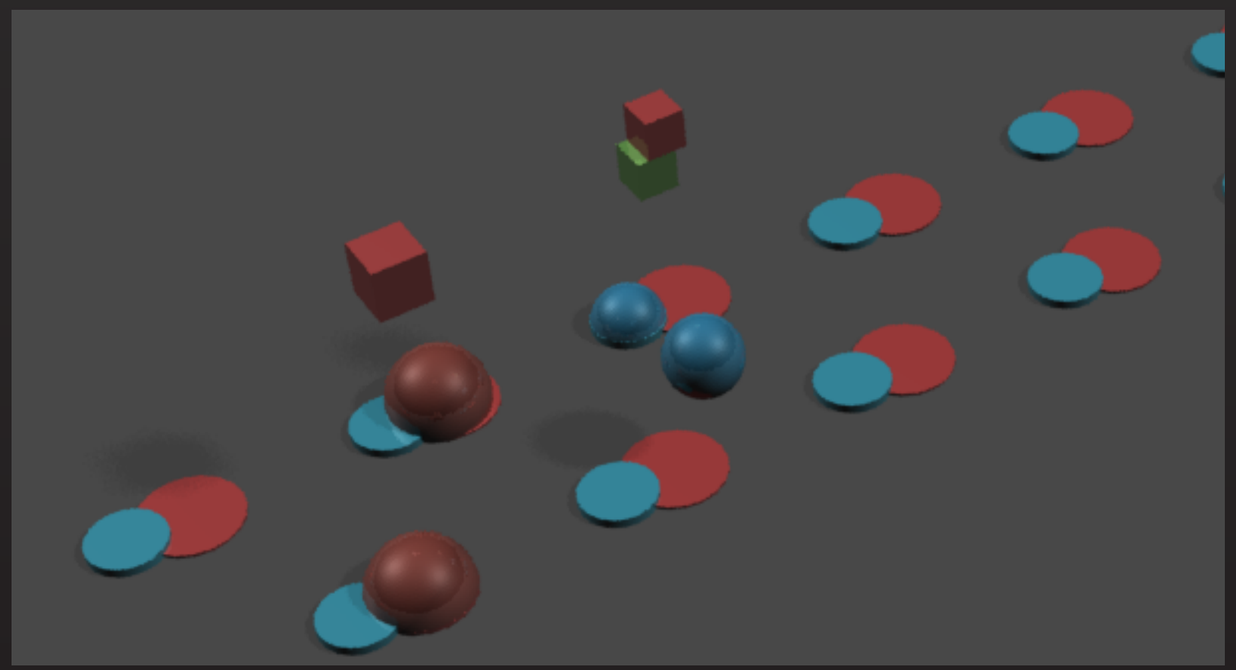
```
55  def createHipsIndicator(trackway):
56      hips, shape = cmds.polyCube(name='hips', width=20, height=20, depth=20)
57      cmds.move(0, 100, 0, hips)
58      references.append(hips)
59
60      shoulders, shape = cmds.polyCube(name='shoulders_locked', width=15, height=15, depth=15)
61      cmds.move(0, 115, trackway.spineLength, shoulders)
62      cmds.parent(shoulders, hips, absolute=True)
63
64      pes = trackway.pes
65      cmds.expression("%s.z = 0.5*abs(%s.z - %s.z) + min(%s.z, %s.z)" % (
66          hips, pes.left, pes.right, pes.left, pes.right))
67
68      return hips
```



Python & Maya Visualization

```
55  def createHipsIndicator(trackway):
56      hips, shape = cmds.polyCube(name='hips', width=20, height=20, depth=20)
57      cmds.move(0, 100, 0, hips)
58      references.append(hips)
59
60      shoulders, shape = cmds.polyCube(name='shoulders_locked', width=15, height=15, depth=15)
61      cmds.move(0, 115, trackway.spineLength, shoulders)
62      cmds.parent(shoulders, hips, absolute=True)
63
64      pes = trackway.pes
65      cmds.expression("%s.z = 0.5*abs(%s.z - %s.z) + min(%s.z, %s.z)" % (
66          hips, pes.left, pes.right, pes.left, pes.right))
67
68      return hips
```

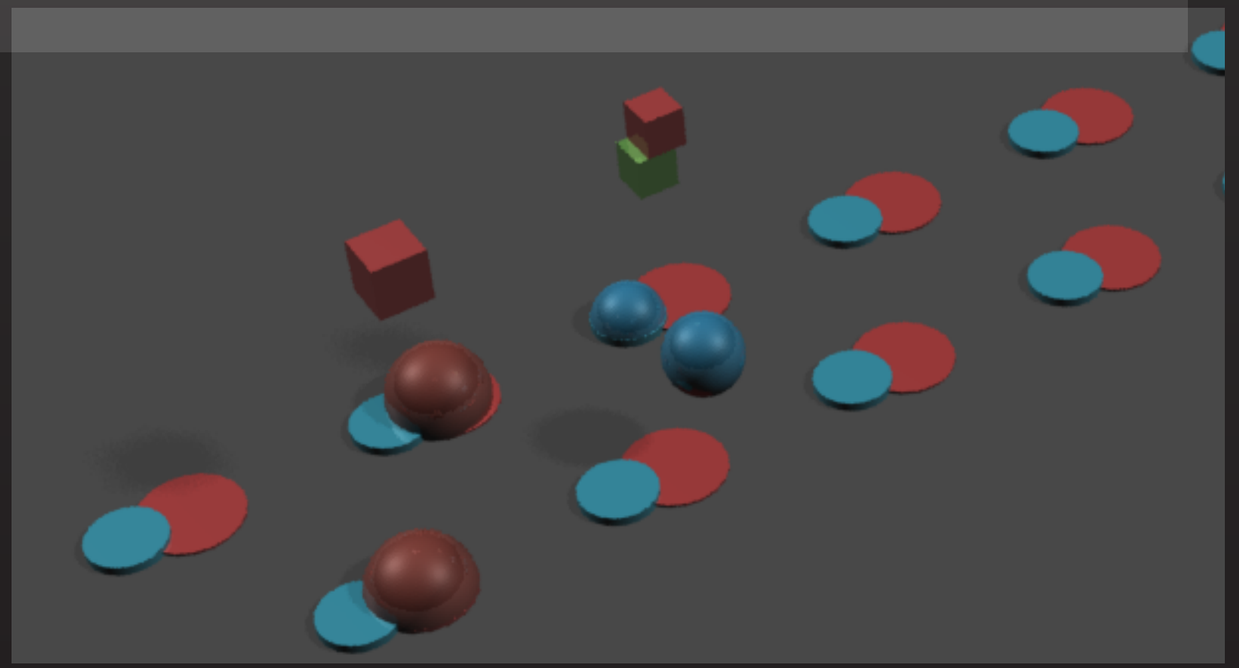
Shoulder Indicator



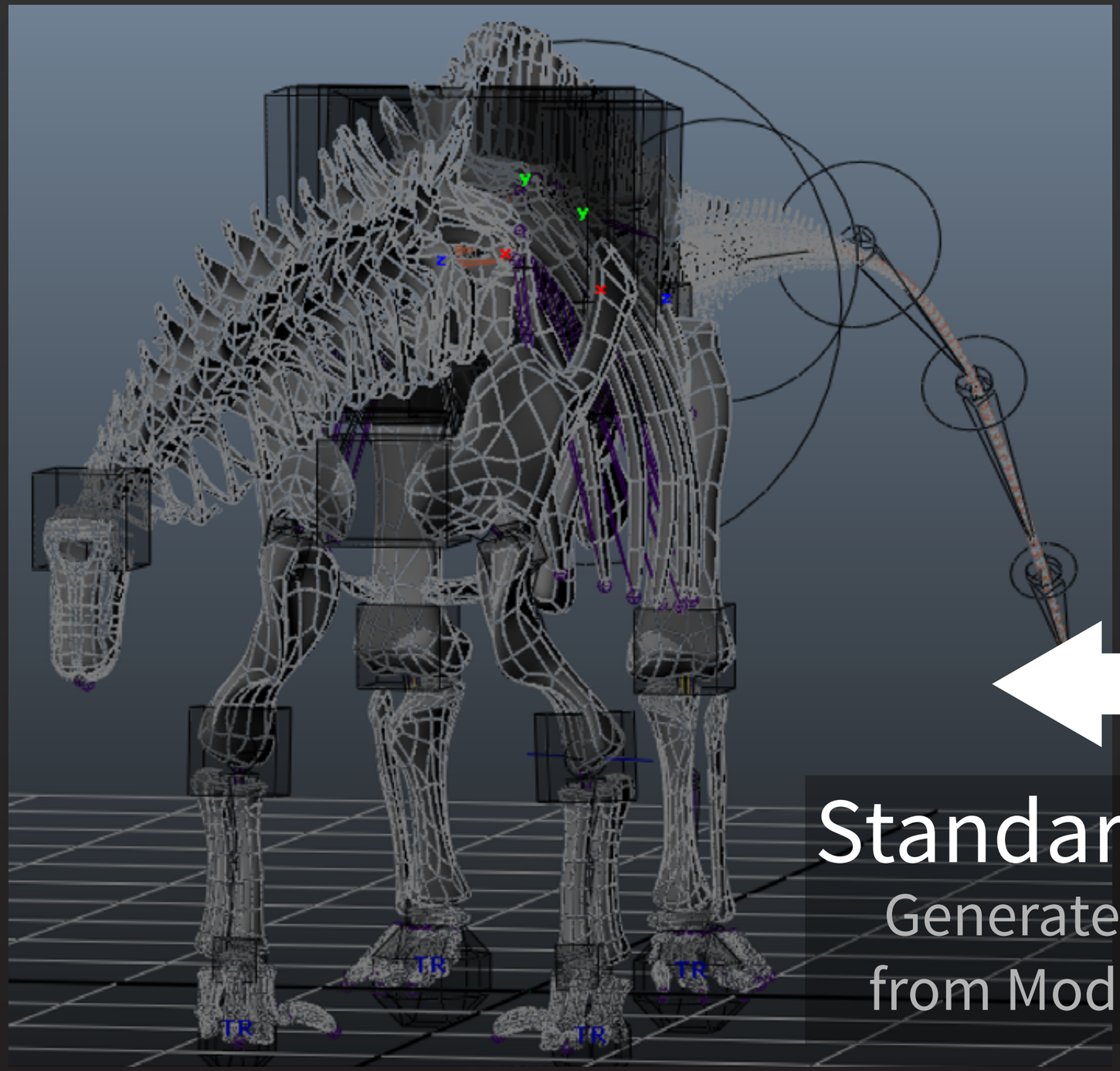
Python & Maya Visualization

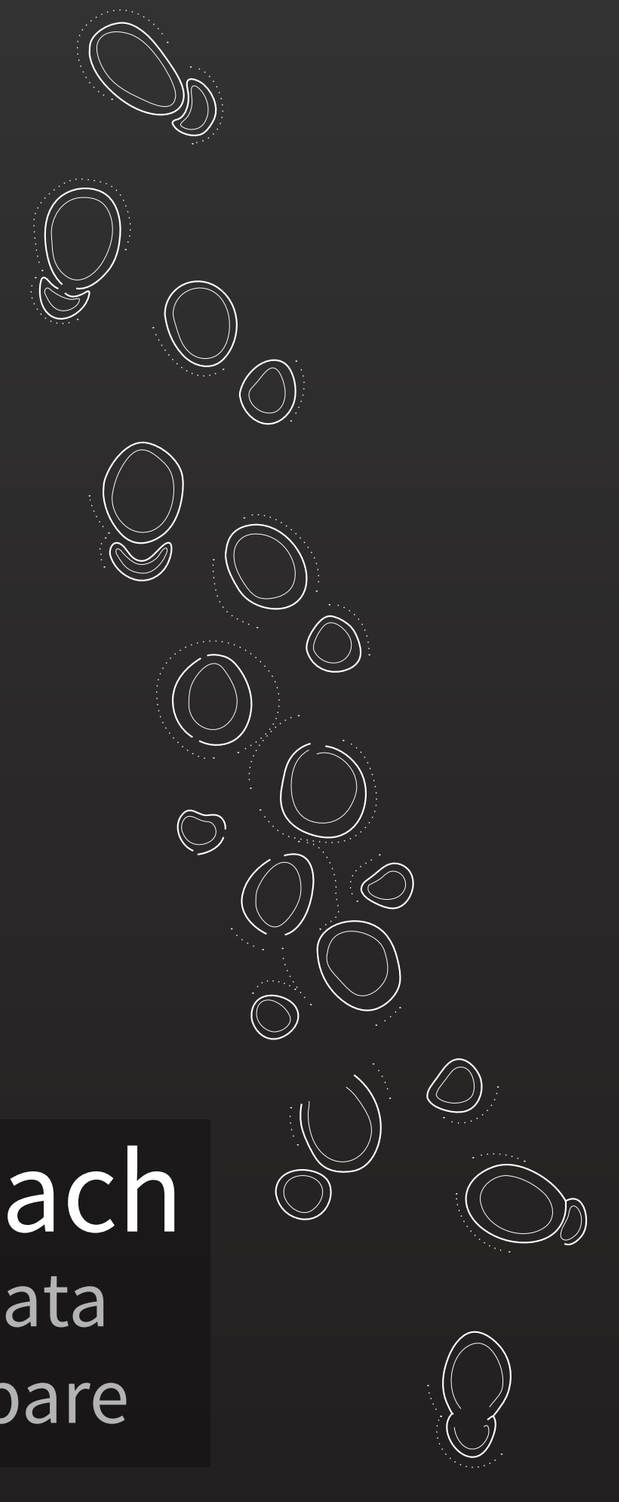
```
55  def createHipsIndicator(trackway):
56      hips, shape = cmds.polyCube(name='hips', width=20, height=20, depth=20)
57      cmds.move(0, 100, 0, hips)
58      references.append(hips)
59
60      shoulders, shape = cmds.polyCube(name='shoulders_locked', width=15, height=15, depth=15)
61      cmds.move(0, 115, trackway.spineLength, shoulders)
62      cmds.parent(shoulders, hips, absolute=True)
63
64      pes = trackway.pes
65      cmds.expression("%s.z = 0.5*abs(%s.z - %s.z) + min(%s.z, %s.z)" % (
66          hips, pes.left, pes.right, pes.left, pes.right))
67
68      return hips
```

Midpoint Expression
(runs every frame update)



Research Approach

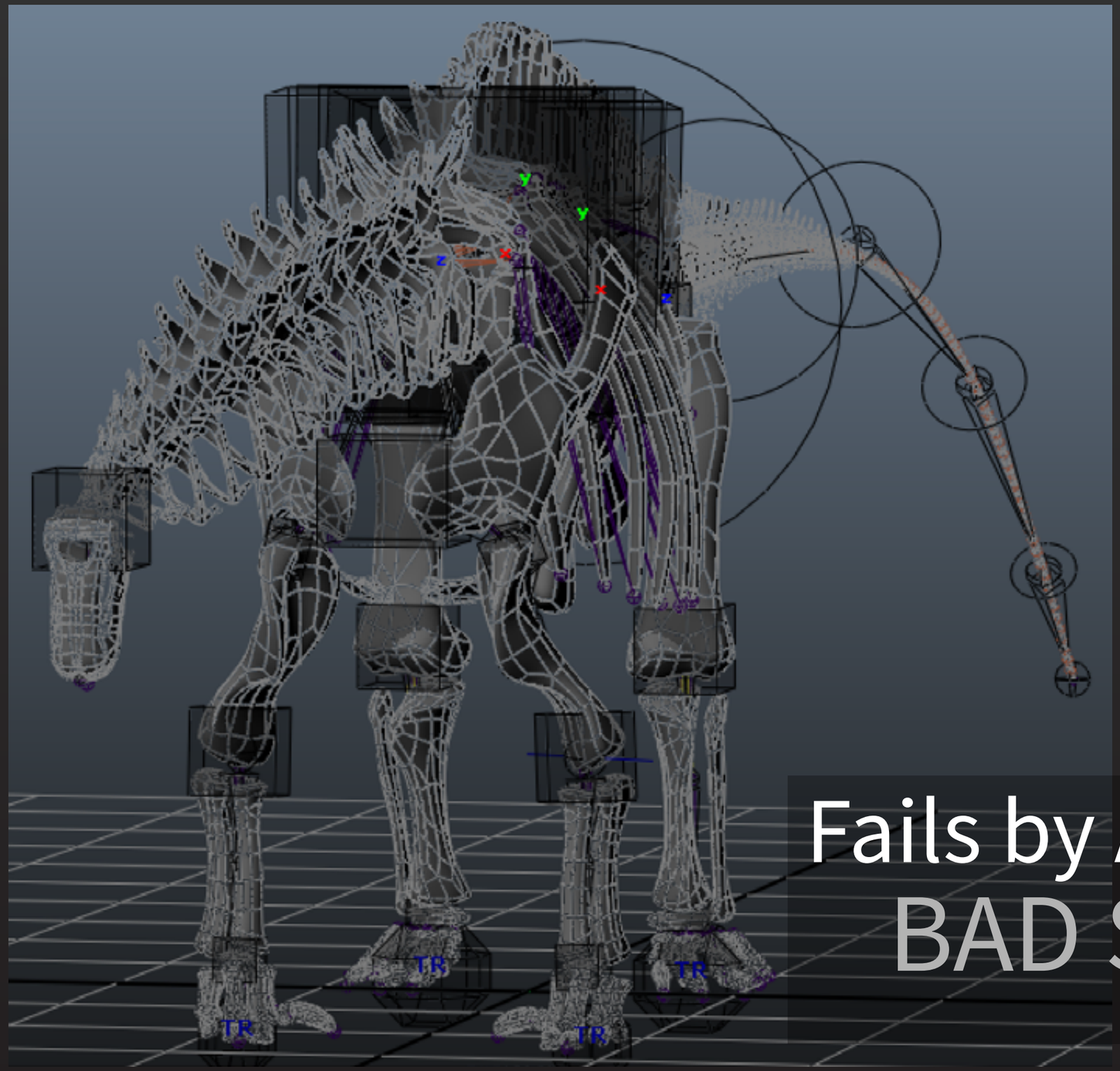




Standard Approach

Generate Synthetic Data
from Model and Compare

Research Approach



Fails by Assumption

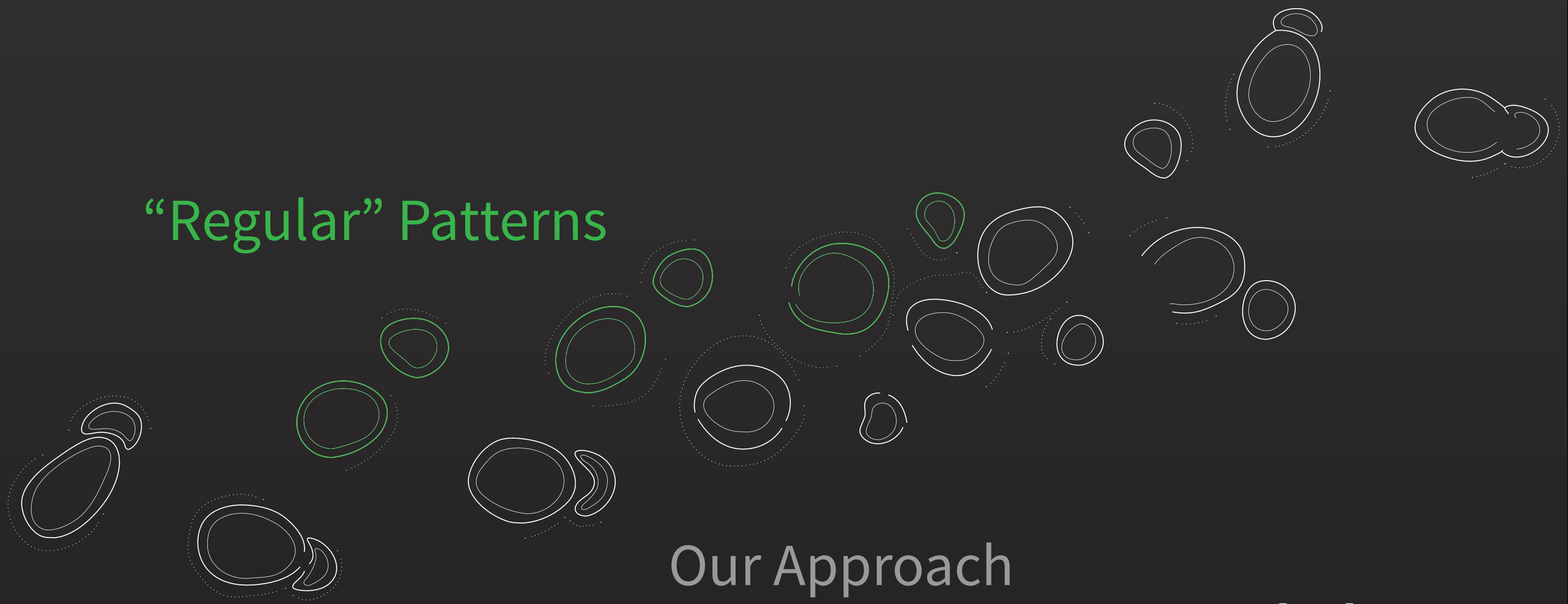
BAD SCIENCE

Research Approach

Our Approach

Data-Driven Modeling

Research Approach



“Regular” Patterns

Our Approach

Data-Driven Modeling

Research Approach

Significant Deviations

“Regular” Patterns

Our Approach

Data-Driven Modeling

Research Approach

Cadence

Trackway Analysis Toolset

/sernst/Cadence

Cadence: Data Input

Field Measurements

Field Drawings

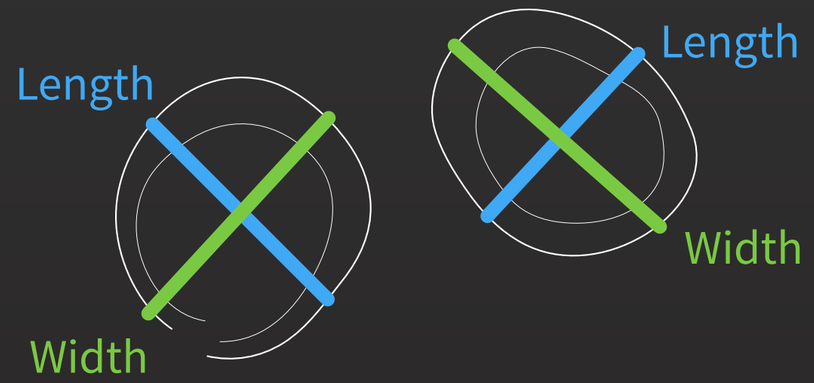
Reference Imagery

Slabs & Casts

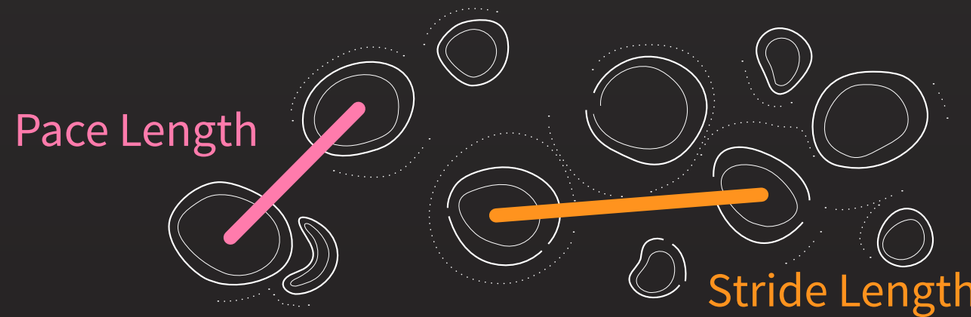
Cadence: Data Input

Field Measurements

- Track Measurements (Local Axes)

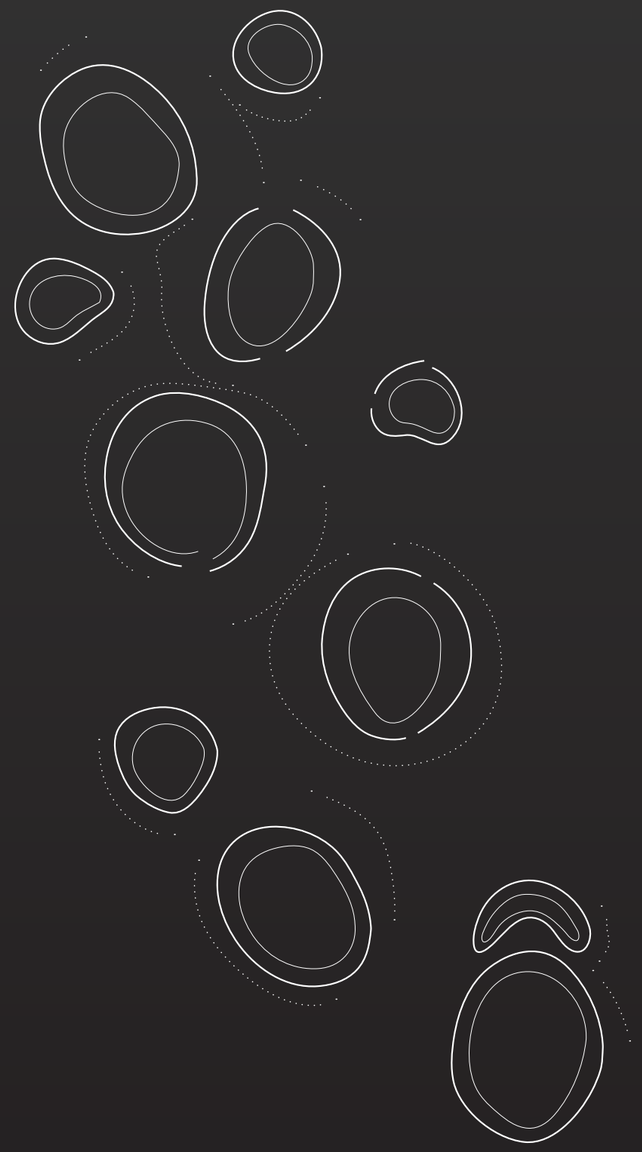


- Track-to-Track Measurements (Relative Axes)



- Good Accuracy & Precision

Cadence: Data Input



Field Drawings

- Absolute Positions & Orientations
- Much Lower Accuracy & Precision
- Loss of Characteristic Structure
- Greater Chance of Data Corruption

Cadence: Data Input

Reference Imagery

- Variable Quality
- Perspective Issues
- Relatively Low Resolution

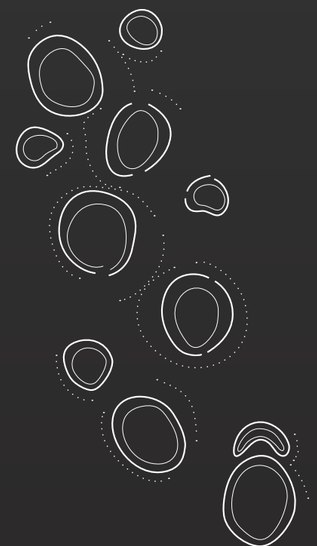
Cadence: Data Input

- “Little” Pieces

- Small Fraction of Tracks Saved

Cadence: Data Input

Field Measurements

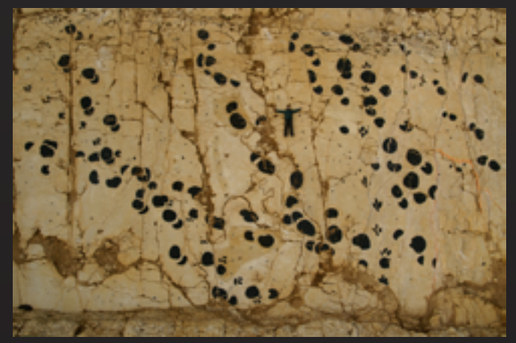


Field Drawings

Cadence Database

Slabs & Casts

Resolve Conflicts



Reference Imagery

Cadence: Input Data

Uncertainty

Variable Quality of Measurement

Cadence: Database

SQLAlchemy

- Object Relational Mapper (ORM)
- Simplifies Database Interaction
- Supports Multiple Relational Databases

Cadence Distributed Database (SQLite)

Cadence: Database

```
 3 import sqlalchemy as sqla
 4
 5 class Track(ModelsBase):
 6
 7     __tablename__   = 'tracks'
 8
 9     i                   = sqla.Column(sqla.Integer,     primary_key=True)
10     uid                 = sqla.Column(sqla.Unicode,     default='')
11     site                = sqla.Column(sqla.Unicode,     default='')
12     year                = sqla.Column(sqla.Unicode,     default='')
13     level               = sqla.Column(sqla.Unicode,     default='')
14     sector              = sqla.Column(sqla.Unicode,     default='')
15     trackwayType        = sqla.Column(sqla.Unicode,     default='')
16     trackwayNumber      = sqla.Column(sqla.Unicode,     default='')
17     number              = sqla.Column(sqla.Unicode,     default='')
18
19     next                = sqla.Column(sqla.Unicode,     default='')
20     left                = sqla.Column(sqla.Boolean,     default=True)
21     pes                 = sqla.Column(sqla.Boolean,     default=True)
22
23     width               = sqla.Column(sqla.Float,       default=0.0)
24     widthUncertainty    = sqla.Column(sqla.Float,       default=3.0)
25     length              = sqla.Column(sqla.Float,       default=0.0)
26     lengthUncertainty   = sqla.Column(sqla.Float,       default=3.0)
27     rotation            = sqla.Column(sqla.Float,       default=0.0)
28     rotationUncertainty = sqla.Column(sqla.Float,       default=5.0)
29     x                   = sqla.Column(sqla.Float,       default=0.0)
30     z                   = sqla.Column(sqla.Float,       default=0.0)
31
32     fieldWidth          = sqla.Column(sqla.Float,       default=0.0)
33     fieldLength         = sqla.Column(sqla.Float,       default=0.0)
34     fieldRotation       = sqla.Column(sqla.Float,       default=0.0)
35
36     def getPreviousTrack(self):
37         model = self.__class__
38
39         return self.mySession.query(model).filter(model.next == self.uid).first()
40
41     def getNextTrack(self):
42         model = self.__class__
43         return self.mySession.query(self).filter(model.uid == self.next).first()
```

Define Schema

Cadence: Database

```python
import sqlalchemy as sqla

class Track(ModelsBase):

    __tablename__   = 'tracks'

    i                   = sqla.Column(sqla.Integer,     primary_key=True)
    uid                 = sqla.Column(sqla.Unicode,     default='')
    site                = sqla.Column(sqla.Unicode,     default='')
    year                = sqla.Column(sqla.Unicode,     default='')
    level               = sqla.Column(sqla.Unicode,     default='')
    sector              = sqla.Column(sqla.Unicode,     default='')
    trackwayType        = sqla.Column(sqla.Unicode,     default='')
    trackwayNumber      = sqla.Column(sqla.Unicode,     default='')
    number              = sqla.Column(sqla.Unicode,     default='')

    next                = sqla.Column(sqla.Unicode,     default='')
    left                = sqla.Column(sqla.Boolean,     default=True)
    pes                 = sqla.Column(sqla.Boolean,     default=True)

    width               = sqla.Column(sqla.Float,       default=0.0)
    widthUncertainty    = sqla.Column(sqla.Float,       default=3.0)
    length              = sqla.Column(sqla.Float,       default=0.0)
    lengthUncertainty   = sqla.Column(sqla.Float,       default=3.0)
    rotation            = sqla.Column(sqla.Float,       default=0.0)
    rotationUncertainty = sqla.Column(sqla.Float,       default=5.0)
    x                   = sqla.Column(sqla.Float,       default=0.0)
    z                   = sqla.Column(sqla.Float,       default=0.0)

    fieldWidth          = sqla.Column(sqla.Float,       default=0.0)
    fieldLength         = sqla.Column(sqla.Float,       default=0.0)
    fieldRotation       = sqla.Column(sqla.Float,       default=0.0)

    def getPreviousTrack(self):
        model = self.__class__

        return self.mySession.query(model).filter(model.next == self.uid).first()

    def getNextTrack(self):
        model = self.__class__
        return self.mySession.query(self).filter(model.uid == self.next).first()
```

Separate Values for:

1. Field Measurements
2. Drawing

Cadence: Database

Specify Table

```
 3 import sqlalchemy as sqla
 4
 5 class Track(ModelsBase):
 6
 7     __tablename__   = 'tracks'
 8
 9     i                   = sqla.Column(sqla.Integer,     primary_key=True)
10     uid                 = sqla.Column(sqla.Unicode,     default='')
11     site                = sqla.Column(sqla.Unicode,     default='')
12     year                = sqla.Column(sqla.Unicode,     default='')
13     level               = sqla.Column(sqla.Unicode,     default='')
14     sector              = sqla.Column(sqla.Unicode,     default='')
15     trackwayType        = sqla.Column(sqla.Unicode,     default='')
16     trackwayNumber      = sqla.Column(sqla.Unicode,     default='')
17     number              = sqla.Column(sqla.Unicode,     default='')
18
19     next                = sqla.Column(sqla.Unicode,     default='')
20     left                = sqla.Column(sqla.Boolean,     default=True)
21     pes                 = sqla.Column(sqla.Boolean,     default=True)
22
23     width               = sqla.Column(sqla.Float,       default=0.0)
24     widthUncertainty    = sqla.Column(sqla.Float,       default=3.0)
25     length              = sqla.Column(sqla.Float,       default=0.0)
26     lengthUncertainty   = sqla.Column(sqla.Float,       default=3.0)
27     rotation            = sqla.Column(sqla.Float,       default=0.0)
28     rotationUncertainty = sqla.Column(sqla.Float,       default=5.0)
29     x                   = sqla.Column(sqla.Float,       default=0.0)
30     z                   = sqla.Column(sqla.Float,       default=0.0)
31
32     fieldWidth          = sqla.Column(sqla.Float,       default=0.0)
33     fieldLength         = sqla.Column(sqla.Float,       default=0.0)
34     fieldRotation       = sqla.Column(sqla.Float,       default=0.0)
35
36     def getPreviousTrack(self):
37         model = self.__class__
38
39         return self.mySession.query(model).filter(model.next == self.uid).first()
40
41     def getNextTrack(self):
42         model = self.__class__
43         return self.mySession.query(self).filter(model.uid == self.next).first()
```

Cadence: Database

```
 3 import sqlalchemy as sqla
 4
 5 class Track(ModelsBase):
 6
 7     __tablename__   = 'tracks'
 8
 9     i                   = sqla.Column(sqla.Integer,     primary_key=True)
10     uid                 = sqla.Column(sqla.Unicode,     default='')
11     site                = sqla.Column(sqla.Unicode,     default='')
12     year                = sqla.Column(sqla.Unicode,     default='')
13     level               = sqla.Column(sqla.Unicode,     default='')
14     sector              = sqla.Column(sqla.Unicode,     default='')
15     trackwayType        = sqla.Column(sqla.Unicode,     default='')
16     trackwayNumber      = sqla.Column(sqla.Unicode,     default='')
17     number              = sqla.Column(sqla.Unicode,     default='')
18
19     next                = sqla.Column(sqla.Unicode,     default='')
20     left                = sqla.Column(sqla.Boolean,     default=True)
21     pes                 = sqla.Column(sqla.Boolean,     default=True)
22
23     width               = sqla.Column(sqla.Float,       default=0.0)
24     widthUncertainty    = sqla.Column(sqla.Float,       default=3.0)
25     length              = sqla.Column(sqla.Float,       default=0.0)
26     lengthUncertainty   = sqla.Column(sqla.Float,       default=3.0)
27     rotation            = sqla.Column(sqla.Float,       default=0.0)
28     rotationUncertainty = sqla.Column(sqla.Float,       default=5.0)
29     x                   = sqla.Column(sqla.Float,       default=0.0)
30     z                   = sqla.Column(sqla.Float,       default=0.0)
31
32     fieldWidth          = sqla.Column(sqla.Float,       default=0.0)
33     fieldLength         = sqla.Column(sqla.Float,       default=0.0)
34     fieldRotation       = sqla.Column(sqla.Float,       default=0.0)
35
36     def getPreviousTrack(self):
37         model = self.__class__
38
39         return self.mySession.query(model).filter(model.next == self.uid).first()
40
41     def getNextTrack(self):
42         model = self.__class__
43         return self.mySession.query(self).filter(model.uid == self.next).first()
```

Track Iterations

Alembic

A lightweight database migration tool for SQLAlchemy.

Cadence: Database

Alembic Migration Example

```
3 revision = '2f10651e946'
4 down_revision = '28c48155ac2'
5
6 from alembic import op
7 import sqlalchemy as sqla
8
9 def upgrade():
10     op.add_column('tracks', sqla.Column('name', sqla.Unicode, default=''))
11
12 def downgrade():
13     op.drop_column('tracks', sqla.Column('name', sqla.Unicode, default=''))
14
```

Implement Migration Methods

Cadence: Database

Alembic Migration Example

```
3 revision = '2f10651e946'
4 down_revision = '28c48155ac2'
5
6 from alembic import op
7 import sqlalchemy as sqla
8
9 def upgrade():
10     op.add_column('tracks', sqla.Column('name', sqla.Unicode, default=''))
11
12 def downgrade():
13     op.drop_column('tracks', sqla.Column('name', sqla.Unicode, default=''))
14
```

__init__

alembic.upgradeDatabases()

Cadence: Database Population

Read from CSV Spreadsheet File

```
 3 import re
 4 import csv
 5
 6 from cadence.models.tracks import Track
 7
 8 def read(session, path):
 9     with open(path, 'rU') as f:
10         for row in csv.reader(f, delimiter=',', quotechar='"'):
11             entry = dict()
12
13             for column in Reflection.getReflectionList(TrackCsvColumnEnum):
14                 entry[column.name] = row[column.index]
15             fromSpreadsheetEntry(entry, session)
```

Parse CSV File into “List of Lists”

Cadence: Database Population

Read from CSV Spreadsheet File

```
 3 import re
 4 import csv
 5
 6 from cadence.models.tracks import Track
 7
 8 def read(session, path):
 9     with open(path, 'rU') as f:
10         for row in csv.reader(f, delimiter=',', quotechar='"'):
11             entry = dict()
12
13             for column in Reflection.getReflectionList(TrackCsvColumnEnum):
14                 entry[column.name] = row[column.index]
15             fromSpreadsheetEntry(entry, session)
```

Convert Row “List” to “Dictionary”

Cadence: Database Population

Read from CSV Spreadsheet File

```
 3 import re
 4 import csv
 5
 6 from cadence.models.tracks import Track
 7
 8 def read(session, path):
 9     with open(path, 'rU') as f:
10         for row in csv.reader(f, delimiter=',', quotechar='"'):
11             entry = dict()
12
13             for column in Reflection.getReflectionList(TrackCsvColumnEnum):
14                 entry[column.name] = row[column.index]
15             fromSpreadsheetEntry(entry, session)
```

Create Database Entry from Row Data

Cadence: Database Population

```
17 def fromSpreadsheetEntry(row, session):
18
19     track = Track()
20
21     track.site = row.get(TrackCsvColumnEnum.TRACKSITE.name).strip().upper()
22
23     pattern = re.compile('(?P<type>[^0-9\s\t]+)[\s\t]*(?P<number>[^\(\s\t]+)')
24     test = row.get(TrackCsvColumnEnum.TRACKWAY.name).strip().upper()
25     result = pattern.search(test).groupdict()
26     track.trackwayType = result['type'].upper().strip()
27     track.trackwayNumber = result['number'].upper().strip()
28
29     track.fieldWidth = 0.01*float(collapseManusPesProperty(
30         track, row,
31         TCCE.PES_WIDTH, TCCE.PES_WIDTH_GUESS,
32         TCCE.MANUS_WIDTH, TCCE.MANUS_WIDTH_GUESS,
33         '0', IFE.HIGH_WIDTH_UNCERTAINTY, IFE.NO_WIDTH ))
34
35     session.add(track)
```

Create New Track Database Entry

Cadence: Database Population

Copy Values from Spreadsheet Data

```
17 def fromSpreadsheetEntry(row, session):
18
19     track = Track()
20
21     track.site = row.get(TrackCsvColumnEnum.TRACKSITE.name).strip().upper()
22
23     pattern = re.compile('(?P<type>[^0-9\s\t]+)[\s\t]*(?P<number>[^\(\s\t]+)')
24     test = row.get(TrackCsvColumnEnum.TRACKWAY.name).strip().upper()
25     result = pattern.search(test).groupdict()
26     track.trackwayType = result['type'].upper().strip()
27     track.trackwayNumber = result['number'].upper().strip()
28
29     track.fieldWidth = 0.01*float(collapseManusPesProperty(
30         track, row,
31         TCCE.PES_WIDTH, TCCE.PES_WIDTH_GUESS,
32         TCCE.MANUS_WIDTH, TCCE.MANUS_WIDTH_GUESS,
33         '0', IFE.HIGH_WIDTH_UNCERTAINTY, IFE.NO_WIDTH ))
34
35     session.add(track)
```

Cadence: Database Population

```
17 def fromSpreadsheetEntry(row, session):
18
19     track = Track()
20
21     track.site = row.get(TrackCsvColumnEnum.TRACKSITE.name).strip().upper()
22
23     pattern = re.compile('(?P<type>[^0-9\s\t]+)[\s\t]*(?P<number>[^\(\s\t]+)')
24     test = row.get(TrackCsvColumnEnum.TRACKWAY.name).strip().upper()
25     result = pattern.search(test).groupdict()
26     track.trackwayType = result['type'].upper().strip()
27     track.trackwayNumber = result['number'].upper().strip()
28
29     track.fieldWidth = 0.01*float(collapseManusPesProperty(
30         track, row,
31         TCCE.PES_WIDTH, TCCE.PES_WIDTH_GUESS,
32         TCCE.MANUS_WIDTH, TCCE.MANUS_WIDTH_GUESS,
33         '0', IFE.HIGH_WIDTH_UNCERTAINTY, IFE.NO_WIDTH ))
34
35     session.add(track)
```

Parsing & Formatting

Cadence: Database Population

```
17 def fromSpreadsheetEntry(row, session):
18
19     track = Track()
20
21     track.site = row.get(TrackCsvColumnEnum.TRACKSITE.name).strip().upper()
22
23     pattern = re.compile('(?P<type>[^0-9\s\t]+)[\s\t]*(?P<number>[^\(\s\t]+)')
24     test = row.get(TrackCsvColumnEnum.TRACKWAY.name).strip().upper()
25     result = pattern.search(test).groupdict()
26     track.trackwayType = result['type'].upper().strip()
27     track.trackwayNumber = result['number'].upper().strip()
28
29     track.fieldWidth = 0.01*float(collapseManusPesProperty(
30         track, row,
31         TCCE.PES_WIDTH, TCCE.PES_WIDTH_GUESS,
32         TCCE.MANUS_WIDTH, TCCE.MANUS_WIDTH_GUESS,
33         '0', IFE.HIGH_WIDTH_UNCERTAINTY, IFE.NO_WIDTH ))
34
35     session.add(track)
```

Collapsing Fields to Value

Cadence: Database Population

```
17 def fromSpreadsheetEntry(row, session):
18
19     track = Track()
20
21     track.site = row.get(TrackCsvColumnEnum.TRACKSITE.name).strip().upper()
22
23     pattern = re.compile('(?P<type>[^0-9\s\t]+)[\s\t]*(?P<number>[^\(\s\t]+)')
24     test = row.get(TrackCsvColumnEnum.TRACKWAY.name).strip().upper()
25     result = pattern.search(test).groupdict()
26     track.trackwayType = result['type'].upper().strip()
27     track.trackwayNumber = result['number'].upper().strip()
28
29     track.fieldWidth = 0.01*float(collapseManusPesProperty(
30         track, row,
31         TCCE.PES_WIDTH, TCCE.PES_WIDTH_GUESS,
32         TCCE.MANUS_WIDTH, TCCE.MANUS_WIDTH_GUESS,
33         '0', IFE.HIGH_WIDTH_UNCERTAINTY, IFE.NO_WIDTH ))
34
35     session.add(track)
```

Add Entry to Database

Cadence: Database Population

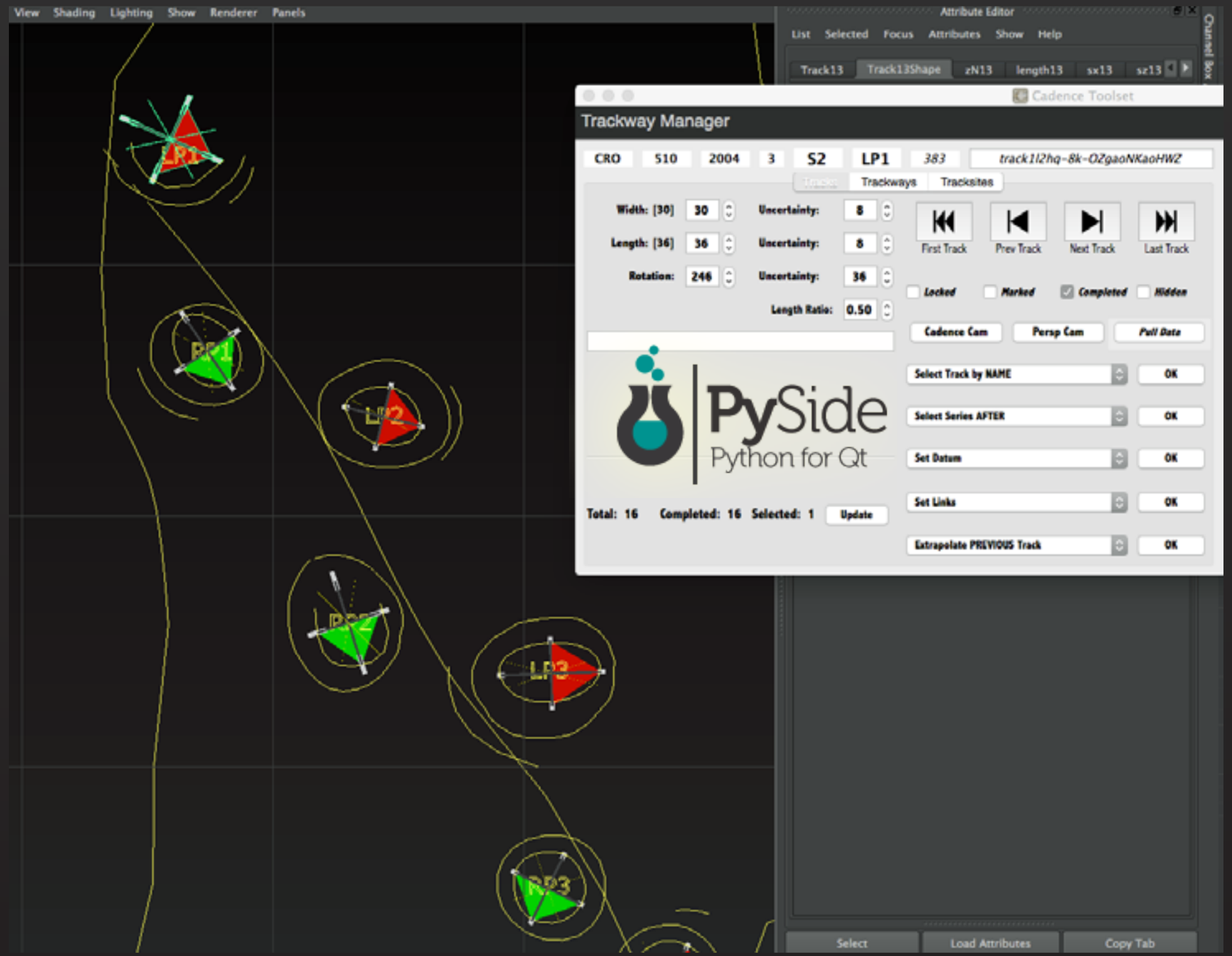
Populate from Field Drawings “Site Maps”

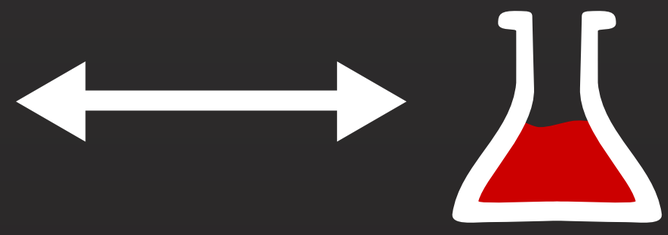
Adobe Illustrator Files

Autodesk Maya

Cadence: Database Population

Populate from Field Drawings “Site Maps”





SQLAlchemy

Data Validation

Field Measurements

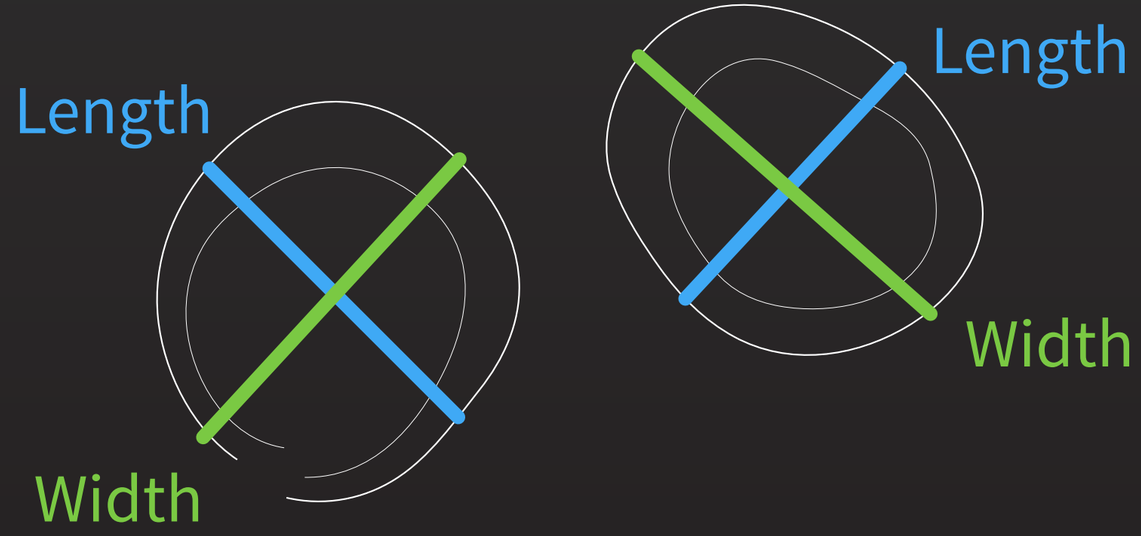
Field Drawings

Coexistence ≠ Agreement

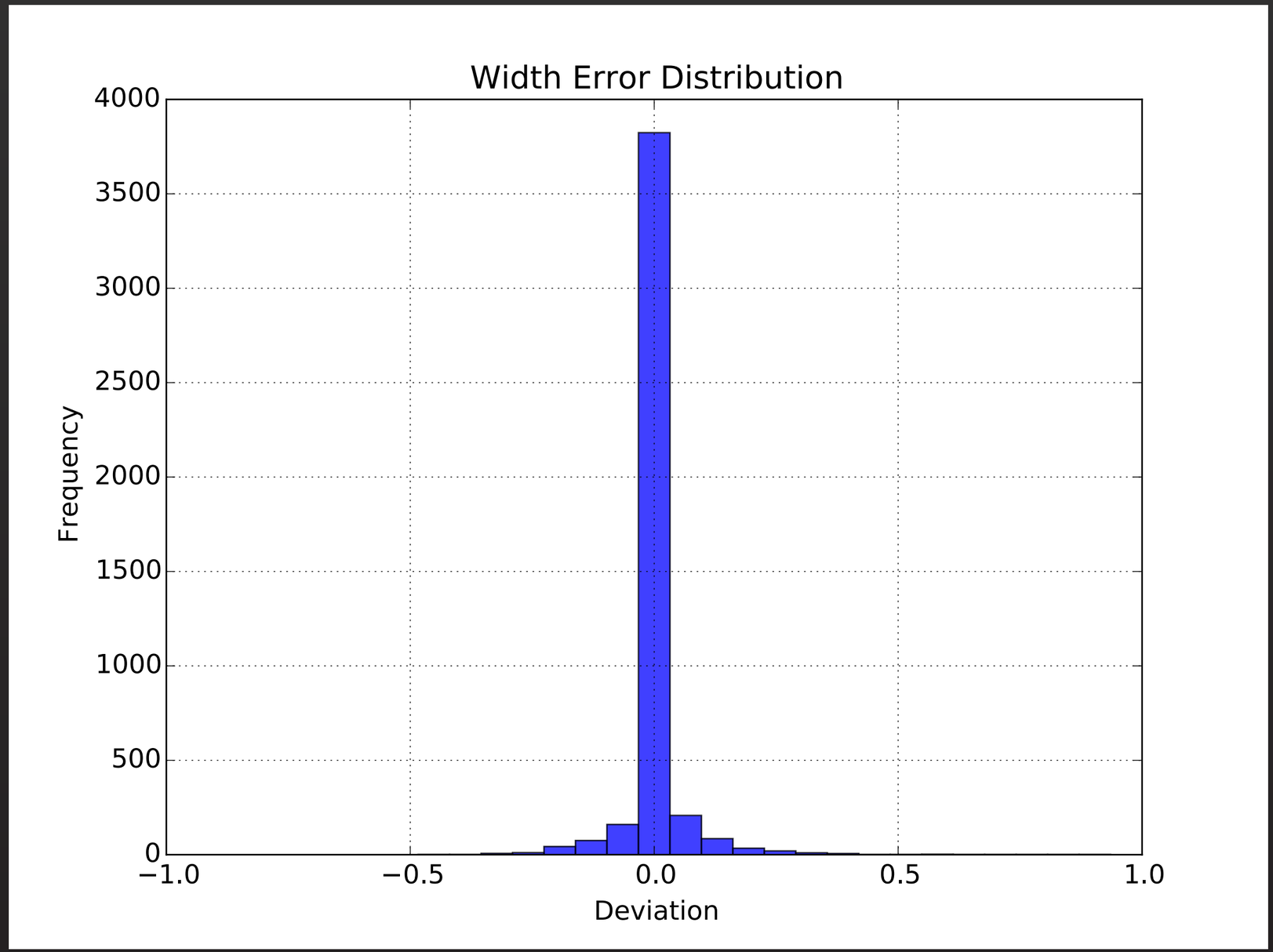
Data Validation

Some Properties:

Direct Comparison

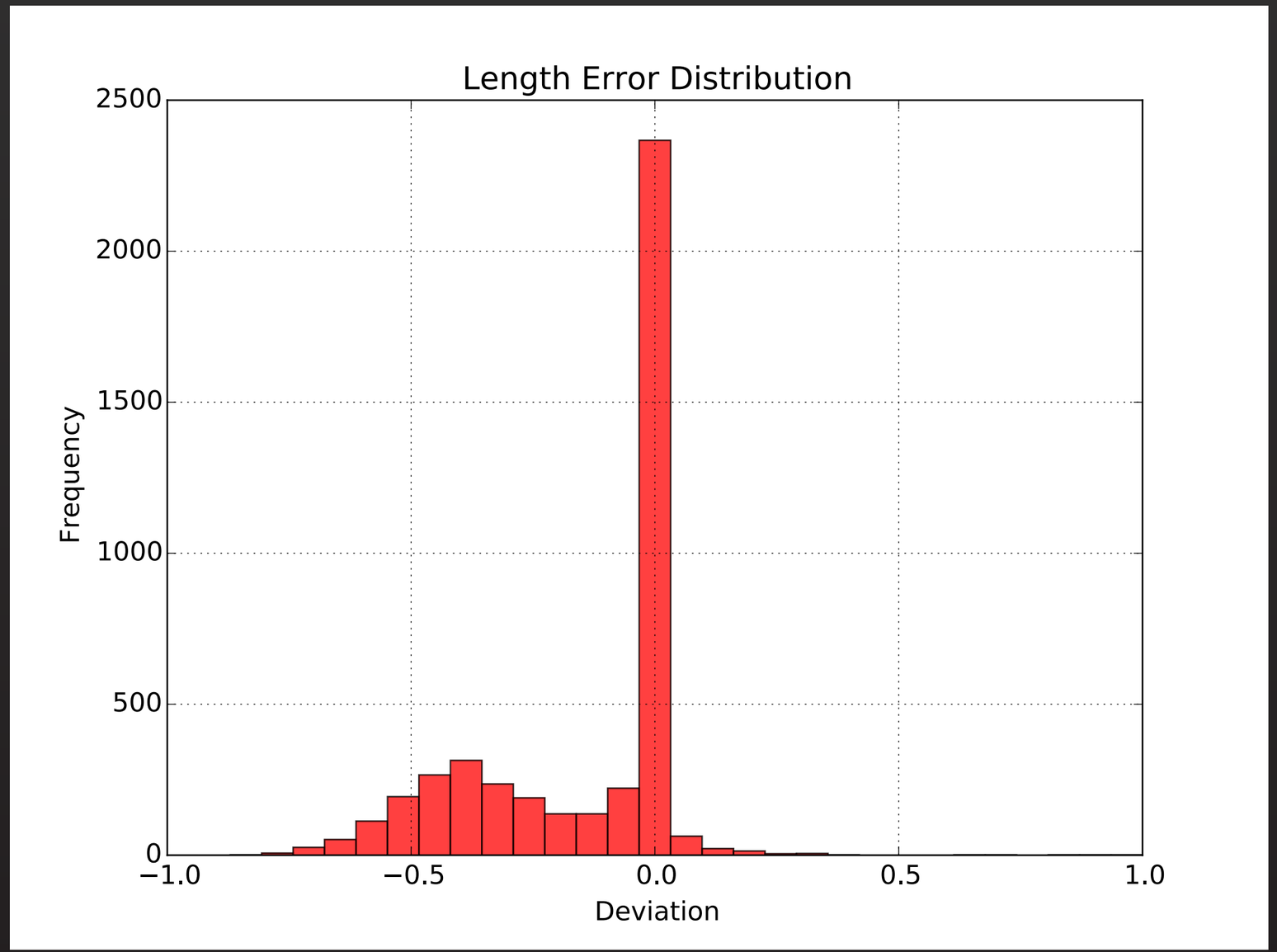


Data Validation



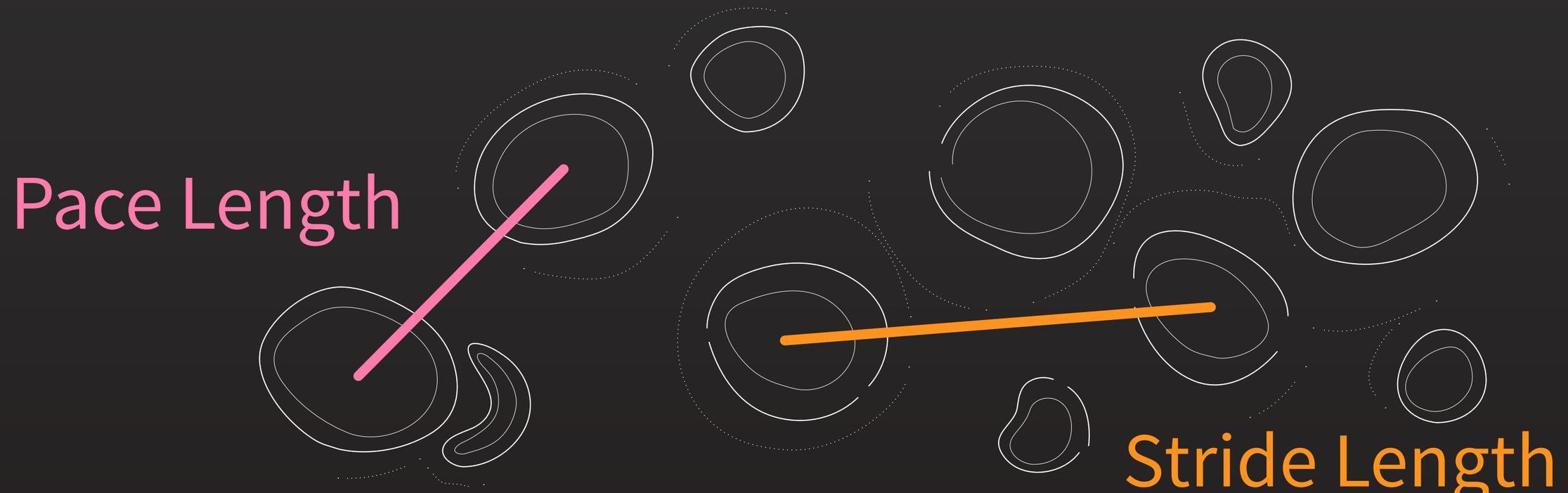
Data Validation

```
 3 import matplotlib.pyplot as plt
 4
 5 def histogram(path, data, bins, xLimits, yLimits, color, isLog, title, xLabel, yLabel, **kwargs):
 6     plt.hist(data, bins, range=xLimits, facecolor=color, log=isLog)
 7     plt.title(title)
 8     plt.xlabel(xLabel)
 9     plt.ylabel(yLabel)
10     plt.xlim(*xLimits)
11     plt.ylim(*yLimits)
12     plt.grid(True)
13
14     figure.savefig(path, **kwargs)
```



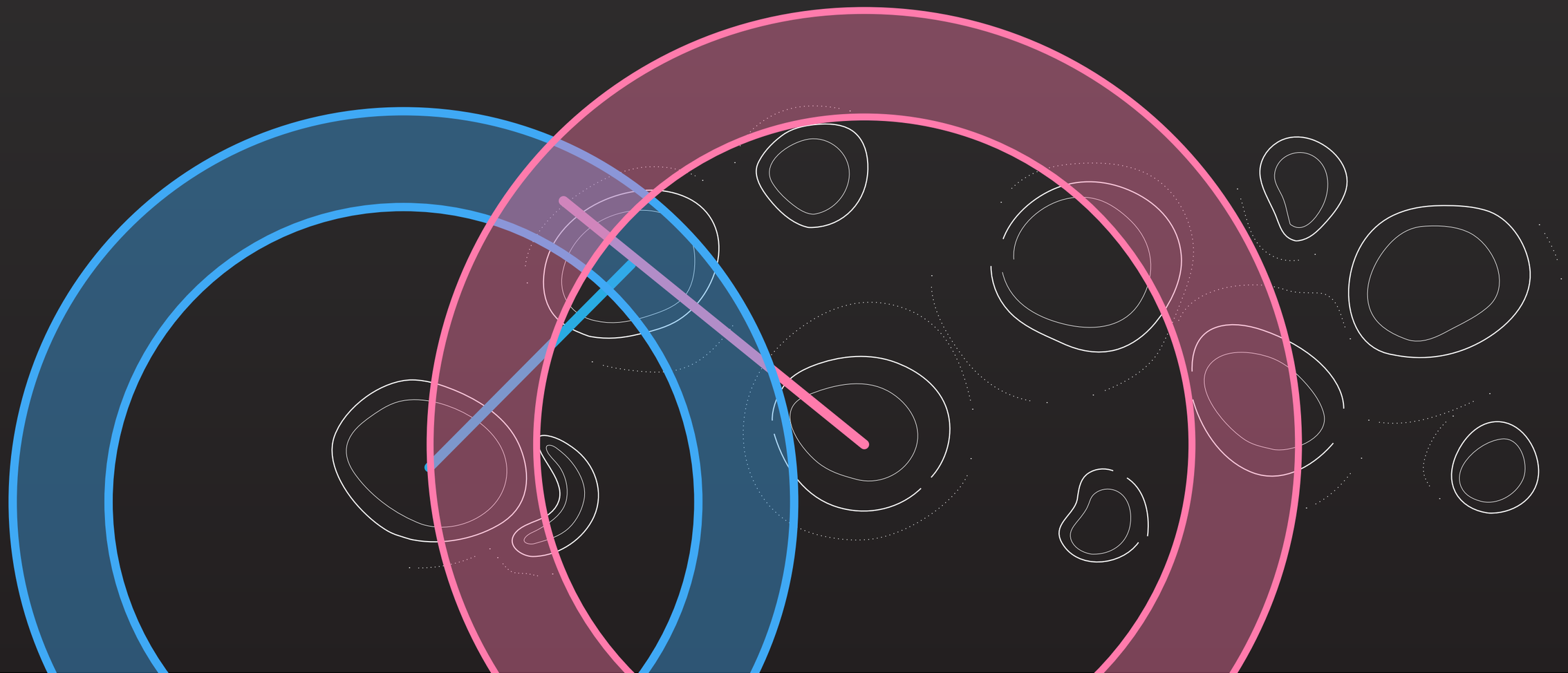
Data Validation

Most Properties:
Indirect Comparison



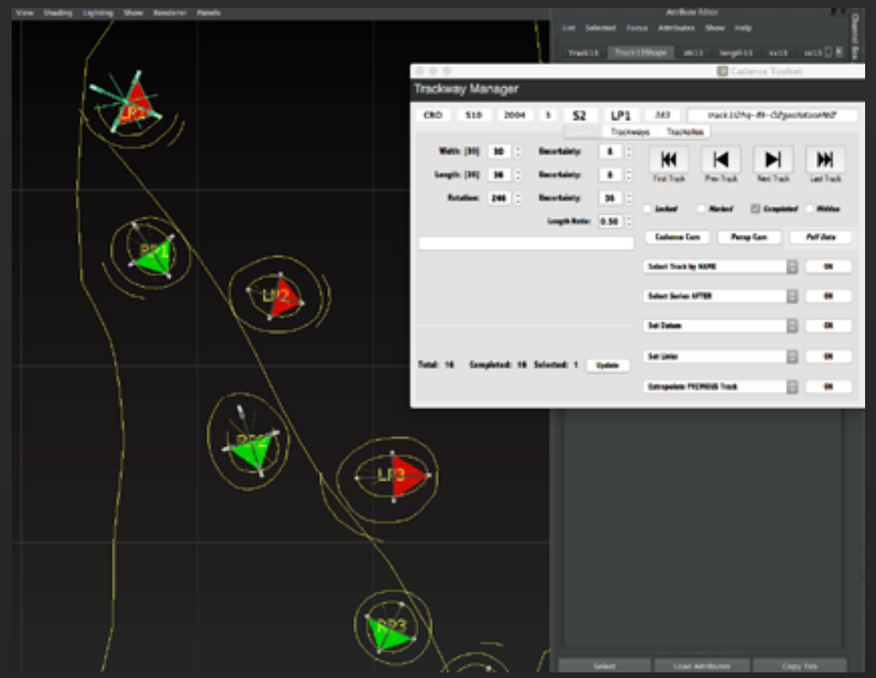
Data Validation

Issue Resolution

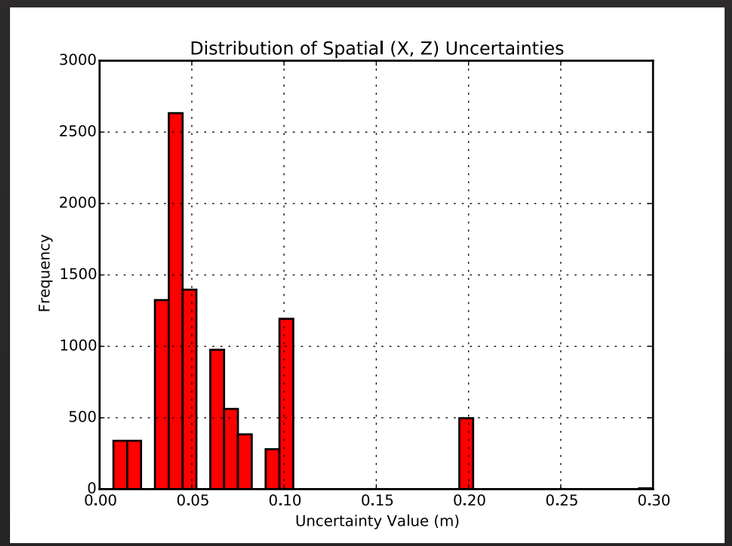


Data Validation

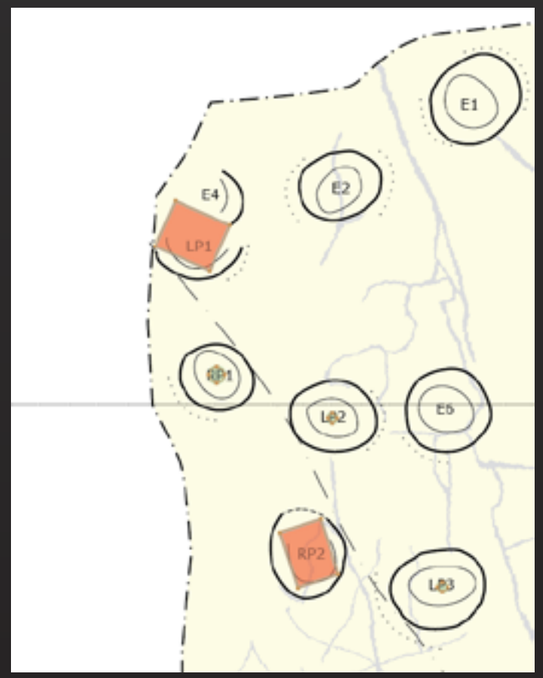
Visualization is Critical for Analysis & Communication



1. Maya



2. Pyplot



3. SVG

Data Validation

Data Validation

SvgWrite Imports

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

Data Validation

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

Create “Drawing”

Data Validation

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

Hole “Mask”

Data Validation

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

Border Circle

Data Validation

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

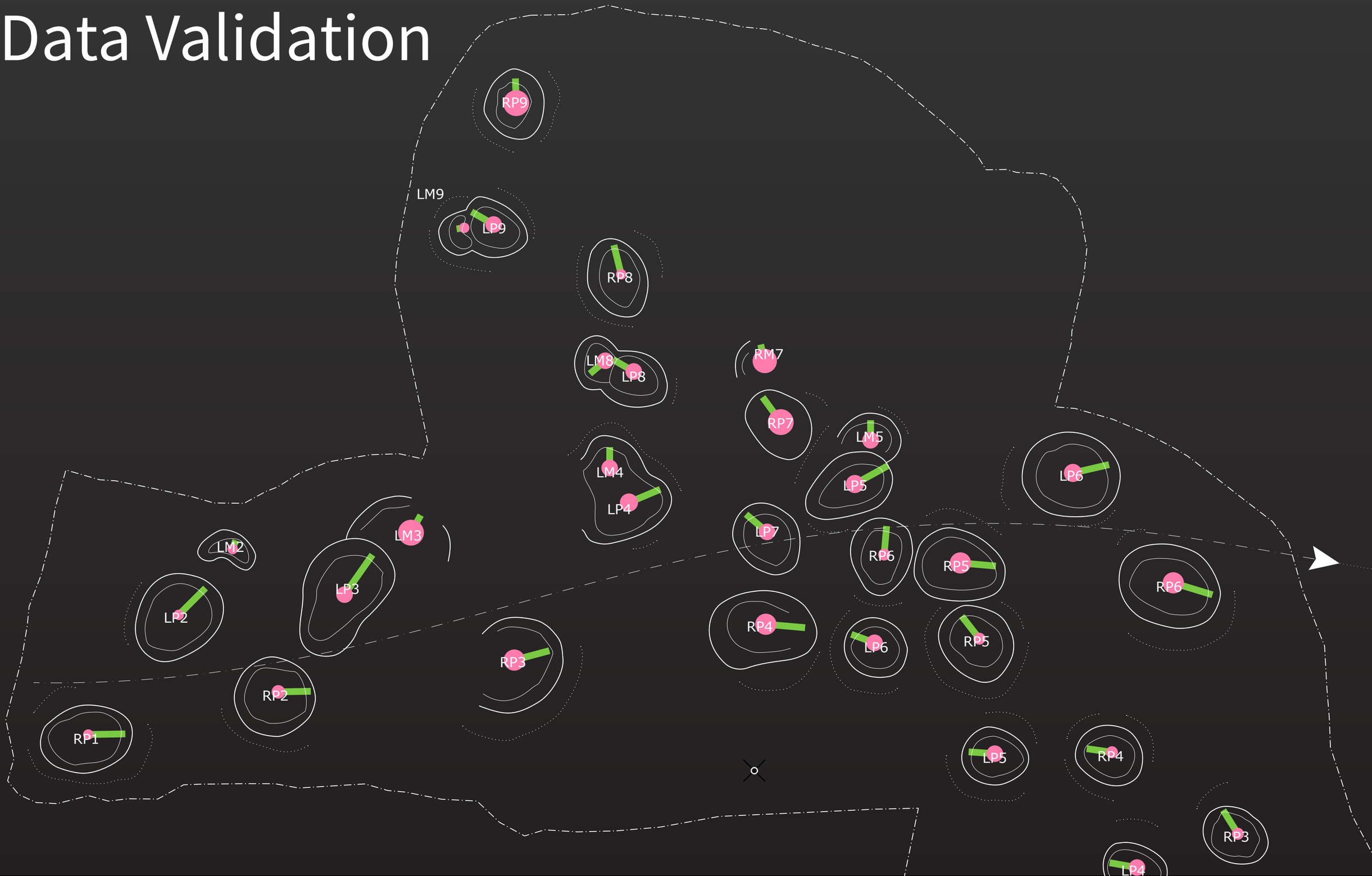
Draw “Ring Segments”

Data Validation

```
 3 import svgwrite
 4 from svgwrite import shapes, masking
 5 from svgwrite import path as svg_path
 6
 7 def drawRing(filename, size, colorPalette, border, center, outerRadius, innerRadius):
 8     drawing = svgwrite.Drawing(filename=filename, size=('%spx' % size[0], '%spx' % size[1]))
 9     drawing.viewbox(0, 0, size[0], size[1])
10
11     mask = drawing.defs.add(masking.Mask(id='hole'))
12     mask.add(shapes.Rect((0, 0), size)).fill('white')
13     mask.add(shapes.Circle(center.toTuple(), r=innerRadius - border)).fill('black')
14
15     drawing.add(shapes.Circle(center.toTuple(), r=outerRadius + border, mask='url(#hole)'))
16
17     start = 0.0
18     for index in range(data):
19         value = data[index]
20         end = start + value
21         color = colorPalette[index % len(colorPalette)]
22
23         points = makeSegmentPath(start, end, outerRadius, innerRadius, center)
24         cmds = [('M', points[0].x, points[0].y)]
25
26         for p in points[1:]:
27             cmds.append(('L', p.x, p.y))
28         cmds.append('Z',)
29
30         drawing.add(svg_path.Path(cmds)).fill(color)
31         start = end
32
33     drawing.save()
```

Save to SVG File

Data Validation



Conclusion

High Performance Computing

Desktop UI

Web Development

Visual FX

Statistical Computing

Numerical Computing

Databases

Breadth & Depth is Very Useful