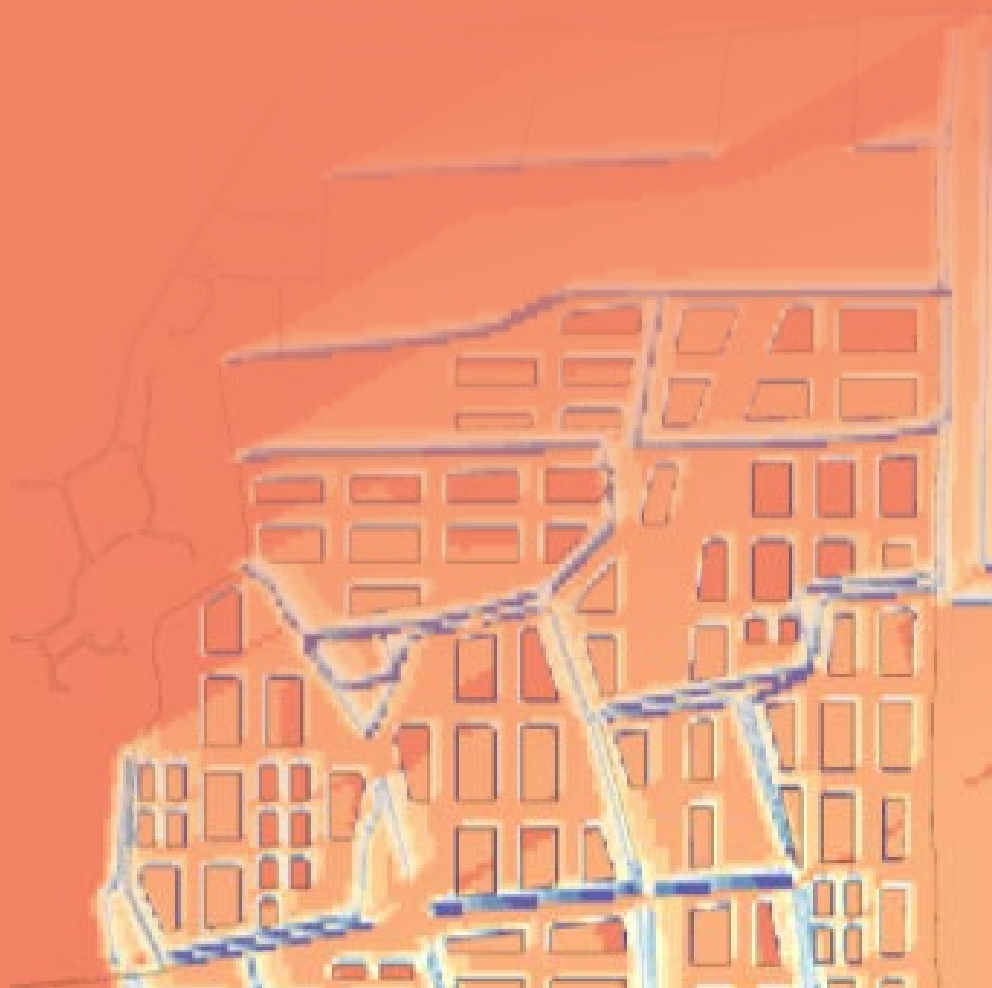
CityAI

 # AI / ML
 # Environment
 # Climate Action
 # Urban Analysis



Incubator Fall 2021
Andrea Imaz (LDN), Victor Okhoya (VNC), Peter Baird (LDN)

Project description

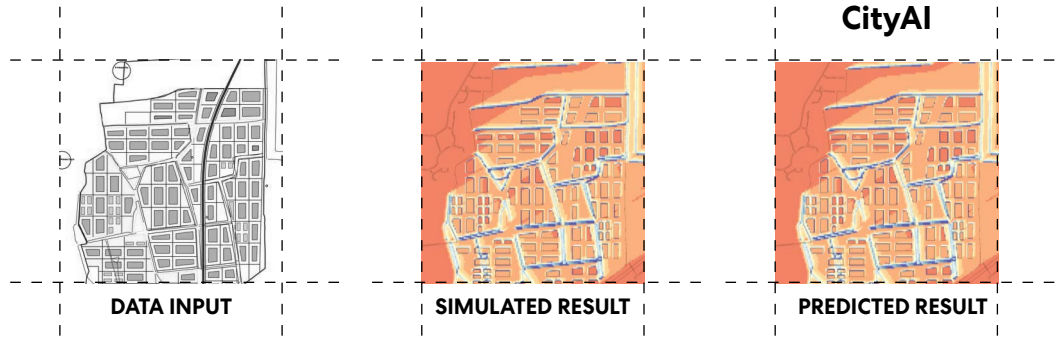
Considering the increasing role of AI in Architecture and Urban Design, in this Innovation Incubator we explore how AI can be leveraged to provide environmental analysis, and trained to assist PW in creating sustainable urban design proposals.

Specifically, we propose to create an **AI model to automate and speed up initial environmental analysis (solar access assessments) targeting optimization of medium/large masterplan projects, from districts to whole cities**.

We believe this is the next innovation step in our practice, moving beyond computational to AI approaches, to facilitate quick assessment of large urban areas, fast-tracking the computational times needed at these scales.

Analysis of large urban scales have historically been challenging due to computational and time expense. However the implementation of AI models opens up the opportunity to work towards reliable simulations, and their implementation in Generative processes where hundreds of design options are tested.

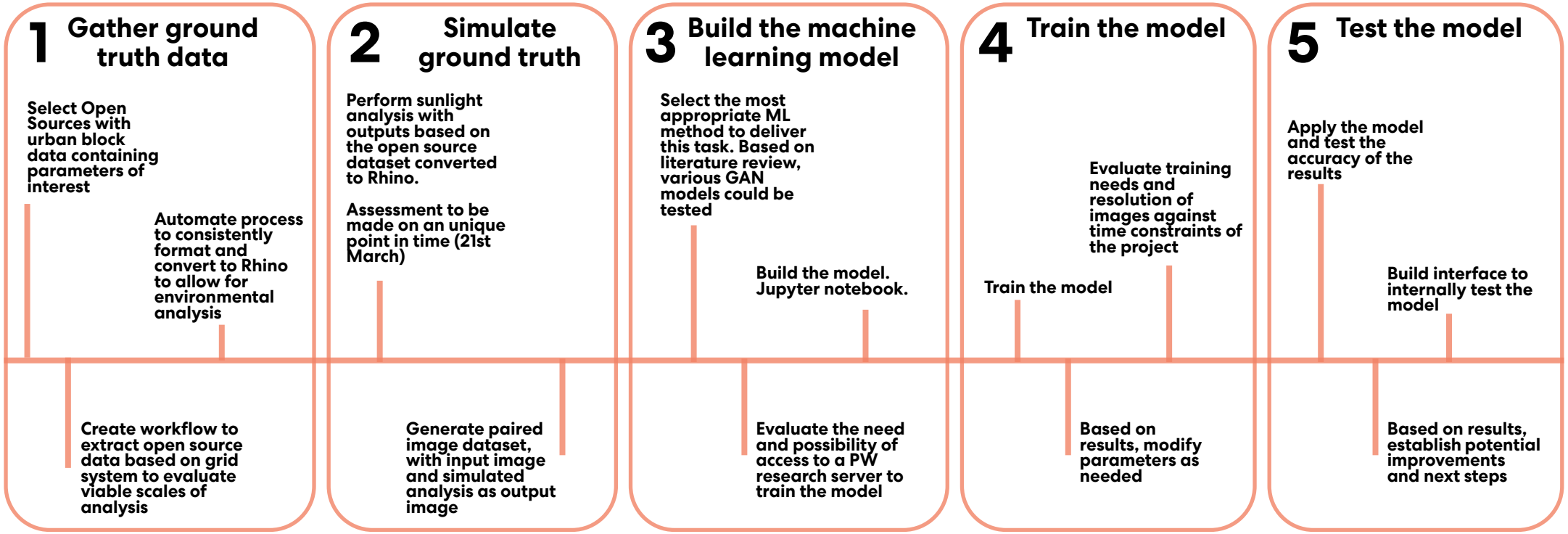
If we can train a model to quickly predict environmental analysis results for large urban scales, how much computing time will be saved?



How “big” can we go?

Could the model be tailored to respond to different geographies? What is the impact of urban morphology “styles”?

Milestones


1
Gather ground truth data
Select Open Sources with urban block data containing parameters of interest
Automate process to consistently format and convert to Rhino to allow for environmental analysis
Create workflow to extract open source data based on grid system to evaluate viable scales of analysis
2
Simulate ground truth
Perform sunlight analysis with outputs based on the open source dataset converted to Rhino.
Assessment to be made on an unique point in time (21st March)
Generate paired image dataset, with input image and simulated analysis as output image
3
Build the machine learning model
Select the most appropriate ML method to deliver this task. Based on literature review, various GAN models could be tested
Build the model. Jupyter notebook.
Evaluate the need and possibility of access to a PW research server to train the model
4
Train the model
Evaluate training needs and resolution of images against time constraints of the project
Train the model
Based on results, modify parameters as needed
5
Test the model
Apply the model and test the accuracy of the results
Build interface to internally test the model
Based on results, establish potential improvements and next steps

1 Gather ground truth data

Gather open source urban block data.

The team downloaded open source data of different cities containing parameters of interest such as footprint and height or number of floors. The layers were obtained on different governmental sites.

Scale of analysis: definition of grid size.

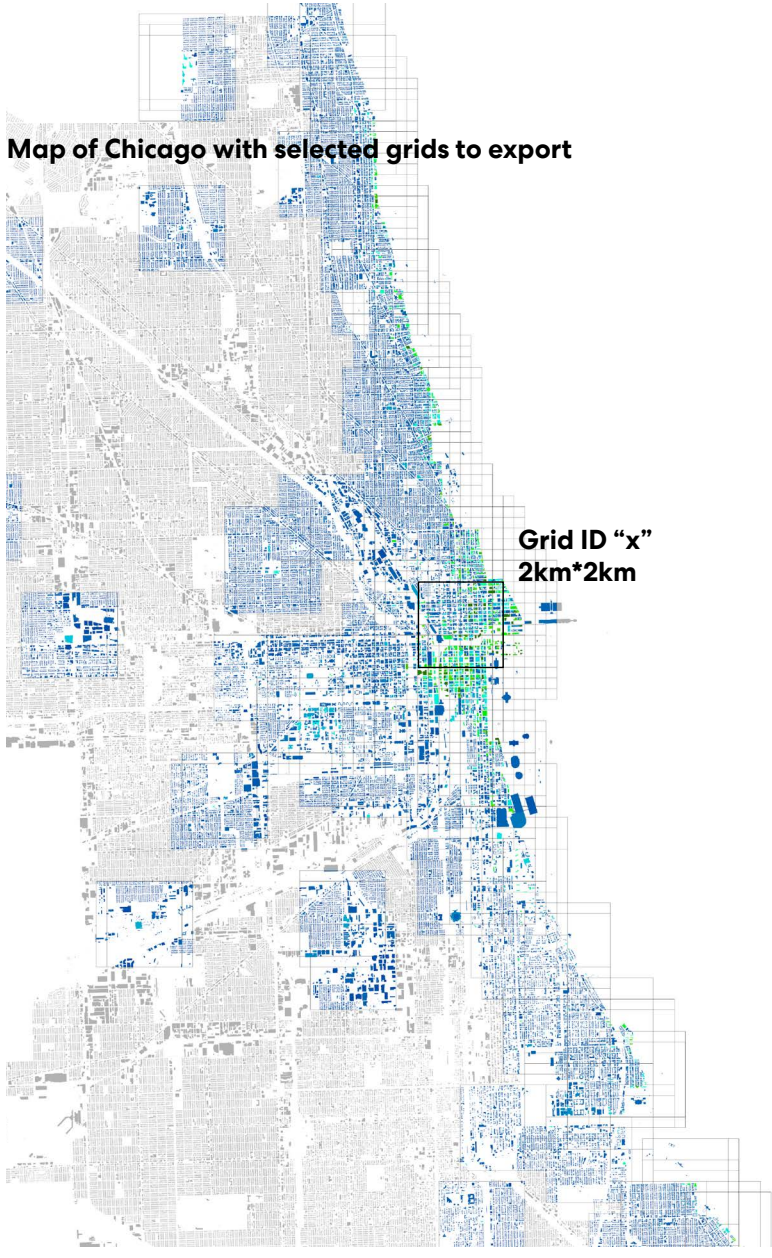
As the goal of the ML model is to provide early performance assessment on urban and masterplanning projects, the team decided to adopt a 2km*2km grid size, mainly for two reasons:

- Considering the size of recent projects, these grid dimensions were able to include project areas together with some of their surrounding context.

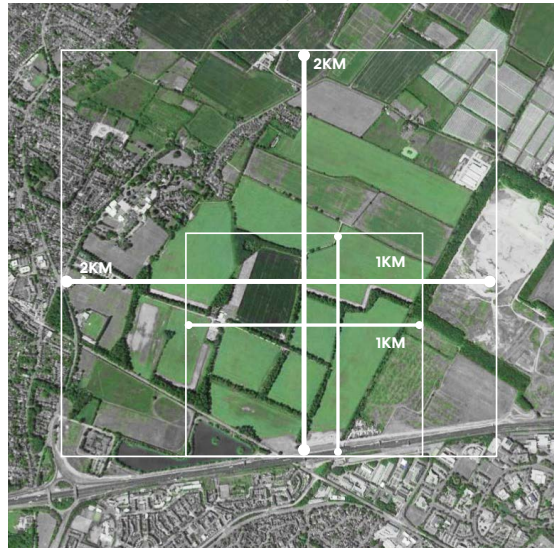
- The analysis of this urban scale would require significant time following the current available resources and, being the model successful, will justify the use of our model for ease of access and time efficiency.

The team selected urban grids with sufficient height variation, so the model could be trained with a wide range of building heights.

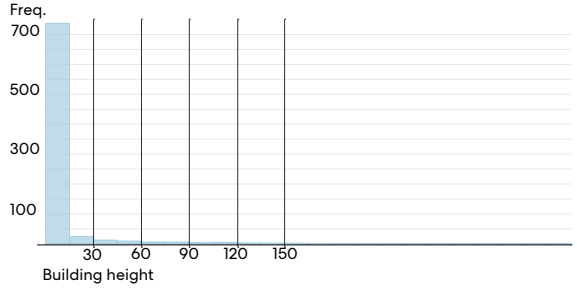
Map of Chicago with selected grids to export



Recent London UD project size reference (CSPN)



Height distribution (Chicago sample)



Building heights were restricited to 150m to avoid outliers that would stretch the color range and concentrate the most frequent heights within a smaller variance.

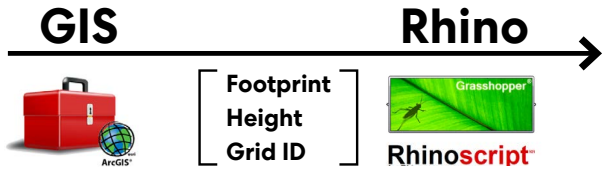
1 Gather ground truth data

Automate process to translate GIS information to Rhino

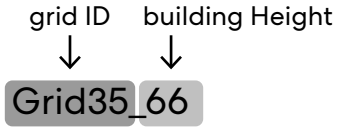
In order to train our model we needed to generate a **paired image dataset**. The input image is the urban morphology with the buildings colored according to their height. The output image is the result of the Solar Access Simulation Analysis on the same urban grid.

To create this images dataset, we transformed the GIS information, including buildings footprint, height and grid ID, into a 3d model so we could perform the Solar Access Analysis within Rhino.

Our AI model is based on a Pix2Pix GAN model which will produce image-to-image translation, learning to map an output image based on an input image. Thus, building heights were translated to a color code range to embed the height information into the input image.



Rhino file with layers structure

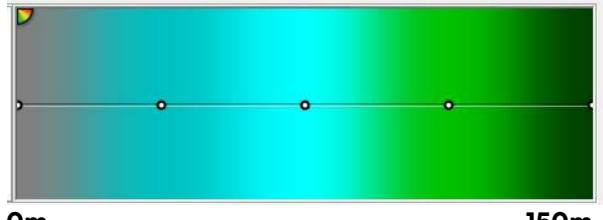


Layers name in Rhino contained all the relevant GIS information. The first part of each layer indicated the Grid ID to which that building belonged to. The second part indicated the height of the building.

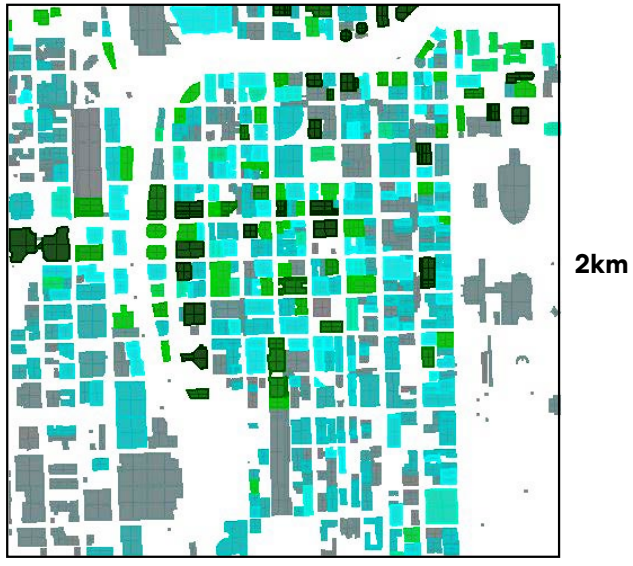
With this information the team was able to create the 3d urban models and perform the Solar Access simulation grid by grid.

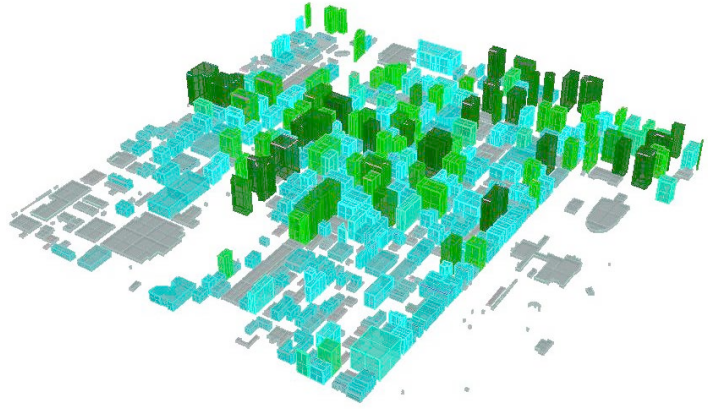
Color range implemented in Rhino

Each height was translated into a color following the below range:



Grid ID "x"





2 Simulate ground truth data

Perform Solar Access Simulation Analysis

The Rhino 3d models produced in the previous step were used to calculate the Simulated Solar Analysis results using the Grasshopper plug-in Ladybug. The Solar Analysis Access simulation ran through the Rhino file to conduct the analysis on each grid independently.

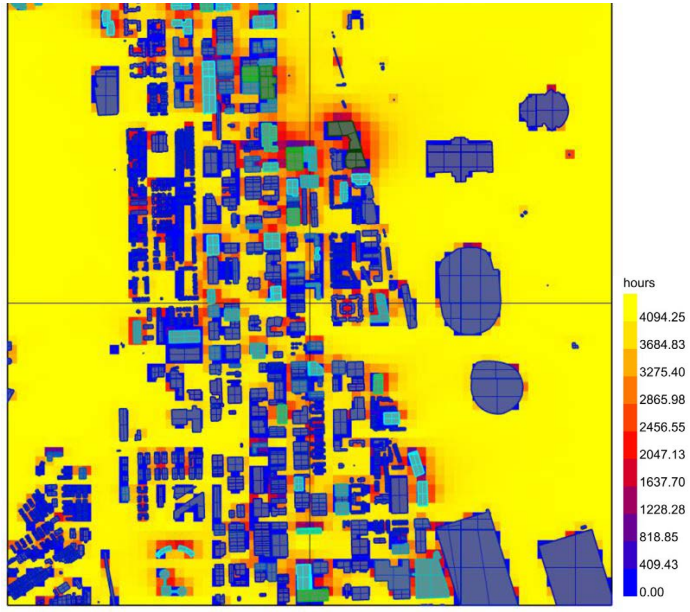
The parameters used in these simulations are as follow:
- Heat map resolution: 5m
- Image Resolution (Rhino) = 600x600px
- Time to simulate in LadyBug = 2min per urban grid
- Date of the year*: 21st September

*Due to time constraints, we used a moment in time analysis. A more detailed simulation throughout the year would have required one hour per urban grid.

Generate paired image dataset

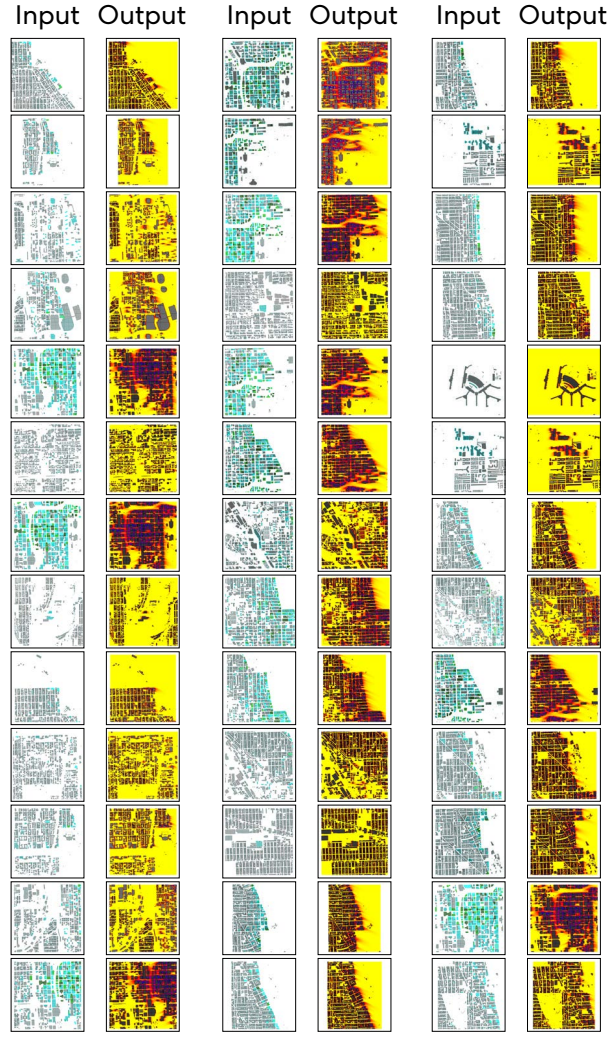
The resulting simulations were recorded as output images within the paired image dataset. A sample of the paired image dataset is shown on the right.

Solar Access Simulation Analysis



The paired image dataset gathered the buildings footprint within each grid coloured by their height as input image, and the simulated Solar Access analysis as the output image. This set of input-output images was used in the ML model for training and testing.

Samples of the paired image dataset generated



3 Building the ML model

Selection of ML model: Pix2Pix GAN

While the ground truth data and the paired image dataset was being gathered the team started to build the model.

The team reviewed previous research working on real-time daylight prediction for building floorplans by Theodore Galanos* and Jason Brownlee's** Pix2Pix GAN model to create a model for this urban solar analysis exercise.

Initially the model was tested with Google Satellite Image dataset*** so the team could get familiarized with the GAN model.

Google Colab Pro with GPU

Given the size of the training dataset, the expected computer hunger and time required, the team opted to use Google Colab Pro, gaining access to faster GPUs to speed the process. The platform also offers more room for data and longer runtimes, with fewer timeouts or disconnections.

* Access to Theodore Galanos project github site: https://github.com/TheodoreGalanos/DaylightGAN

** Pix2Pix GAN model by Jason Borwnlee: https://machinelearningmastery.com/how-to-develop-a-pix2pix-gan-for-image-to-image-translation/

***Dataset obtained from: http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/

Jupyter notebook: building and testing the GAN Model

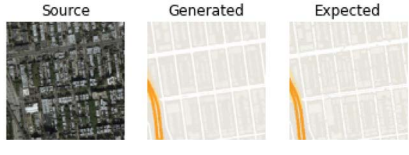
GAN MODEL

In [11]:
```
def define_discriminator(image_shape):
  init = RandomNormal(stddev=0.02)
  in_src_image = Input(shape=image_shape)
  in_target_image = Input(shape=image_shape)
  merged = Concatenate()([in_src_image, in_target_image])
  d = Conv2D(64, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(merged)
  d = LeakyReLU(alpha=0.2)(d)

  d = Conv2D(128, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
  d = BatchNormalization()(d)
  d = LeakyReLU(alpha=0.2)(d)

  d = Conv2D(256, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
  d = BatchNormalization()(d)
  d = LeakyReLU(alpha=0.2)(d)

  d = Conv2D(512, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
  d = BatchNormalization()(d)
  d = LeakyReLU(alpha=0.2)(d)

  d = Conv2D(512, (4,4), padding='same', kernel_initializer=init)(d)
  d = BatchNormalization()(d)
  d = LeakyReLU(alpha=0.2)(d)

  d = Conv2D(1, (4,4), padding='same', kernel_initializer=init)(d)
  patch_out = Activation('sigmoid')(d)

  model = Model([in_src_image, in_target_image], patch_out)
  opt = Adam(lr=0.0002, beta_1=0.5)
  model.compile(loss='binary_crossentropy', optimizer=opt, loss_weights=[0.5])
  return model
```

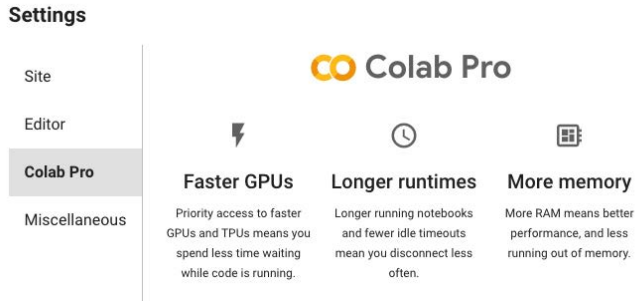
TEST ON TRAINING SET SAMPLE

In [5]:
```
def plot_images(src_img, gen_img, tar_img):
  images = vstack((src_img, gen_img, tar_img))
  images = (images+1)/2.0
  titles = ['Source', 'Generated', 'Expected']

  for i in range(len(images)):
    pyplot.subplot(1, 3, 1+i)
    pyplot.axis('off')
    pyplot.imshow(images[i])
    pyplot.title(titles[i])
  pyplot.show()
```

In [10]:
```
[X1, X2] = load_real_samples('/content/drive/MyDrive/maps/maps_256.npz')
print('Loaded', X1.shape, X2.shape)

model=load_model('/content/drive/MyDrive/maps/model_109600.h5')
ix = randint(0, len(X1), 1)
src_image, tar_image = X1[ix], X2[ix]
gen_image = model.predict(src_image)
plot_images(src_image, gen_image, tar_image)
```



Google Colab Pro with GPU



Colaboratory by Google (Google Colab) is a jupyternotebook environment on the cloud. It enables users to train large scale ML models without investing in powerful machines. As it supports GPU instances, it was the appropiate platform to run this exercise.

4 Training the ML model

Training of the model

Once built, the Pix2Pix GAN model was trained with 241 paired images of 256*256 pixels.

The parameters used in the training process are as follow:
- Training dataset = 241 instances
- Image resolution* (GAN) = 256x256
- Number of training iterations = 24100
- Training time** = 70 minutes (Google Colab with GPU).

241 entries were used to train the ML model. A few instances were set aside for independent evaluation.

*We used low image resolution to speed up the training process given the time limitations of this incubator exercise.

**This is just training time and does not include the time needed to build the model.

Link to the Jupyter Notebook Python file used for this exercise can be found below:

→ **Access to City AI Jupyter notebook: LINK**

Jupyter notebook: City AI training process

LOAD TRAINING IMAGES

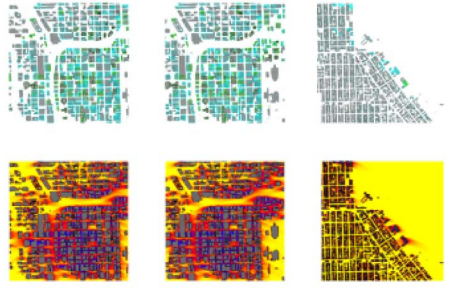
In [4]:
```
def load_images(src_path, tar_path, size=(256,256)):
  src_list, tar_list = list(), list()
  for filename in sorted(listdir(src_path)):
    pixels = load_img(src_path + filename, target_size=size)
    pixels = img_to_array(pixels)
    src_list.append(pixels)

  for filename in sorted(listdir(tar_path)):
    pixels = load_img(tar_path + filename, target_size=size)
    pixels = img_to_array(pixels)
    tar_list.append(pixels)

  return [asarray(src_list), asarray(tar_list)]
```

In [5]:
```
src_path = '/content/drive/MyDrive/_cityai/_input_preprocess/'
tar_path = '/content/drive/MyDrive/_cityai/_output_preprocess/'
[src_images, tar_images] = load_images(src_path,tar_path)
print('Loaded: ',src_images.shape, tar_images.shape)
filename = '/content/drive/MyDrive/_cityai/cityai_256.npz'
savez_compressed(filename, src_images, tar_images)
print('Saved dataset: ', filename)
```

```
Loaded:  (241, 256, 256, 3) (241, 256, 256, 3)
Saved dataset:  /content/drive/MyDrive/_cityai/cityai_256.npz
```

In [6]:
```
from numpy import load
from matplotlib import pyplot
data = load('/content/drive/MyDrive/_cityai/cityai_256.npz')
src_images, tar_images = data['arr_0'],data['arr_1']
print('Loaded: ', src_images.shape, tar_images.shape)
n_samples = 3
for i in range(n_samples):
  pyplot.subplot(2, n_samples, 1+i)
  pyplot.axis('off')
  pyplot.imshow(src_images[i+30].astype('uint8'))
for i in range(n_samples):
  pyplot.subplot(2, n_samples, 1+n_samples+i)
  pyplot.axis('off')
  pyplot.imshow(tar_images[i+30].astype('uint8'))
pyplot.show()
```

```
Loaded:  (241, 256, 256, 3) (241, 256, 256, 3)
```

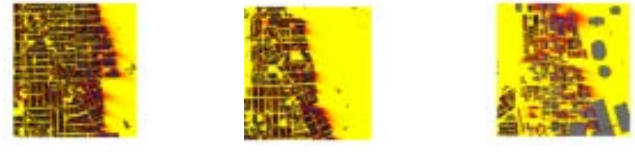


GAN model training process

Input images: initial images with footprint and building height information.

Training process: visualizations of the model performance during the training process.

Generated: results obtained by the model at the end of the training process.



The images above show the evolution of the training process of the ML model, and the final outputs delivered once the process was finished. We validated the results by visual inspection

5 Testing the model

Success and limitations of the model

The process showed a very rapid convergence of the images, demonstrating GAN models are effective for this task. Resolution was reduced to speed up the process, but considering this, we could probably use higher image resolution comfortably and maybe use fewer images or iterations. Further testing would be needed to explore this.

The speed of the process makes it ideal to be integrated within Generative processes where hundreds or thousands of design options need to be assessed.

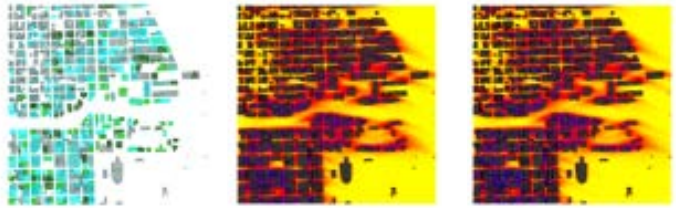
Although we tested the model with new cities (Madrid), we have not done enough investigation to asses its performance in different geographical locations. Our main goal was to prove that the process could work and get reseanable predicted outcomes for Chicago (similar to the simulated ones) but further validation is needed.

Temporary platform

The team created a temporary interface to internally test the model (Python platform Gradio). A video of this interface has been included on the right to give some reference for the development of a permanent platform (see Next Steps section).

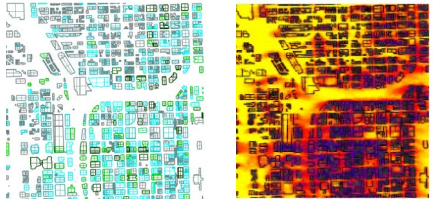
Using Training Set Data

Input image | City AI predicted/ generated | Simulated/ expected



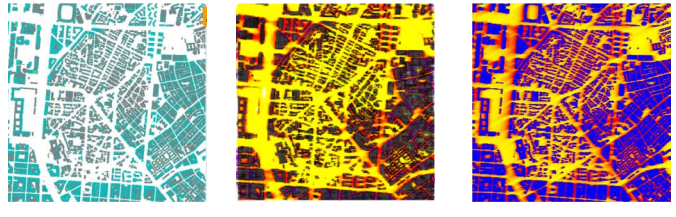
Using Testing Set Data
(not seen by the algorythm before)

Input image | City AI predicted/ generated



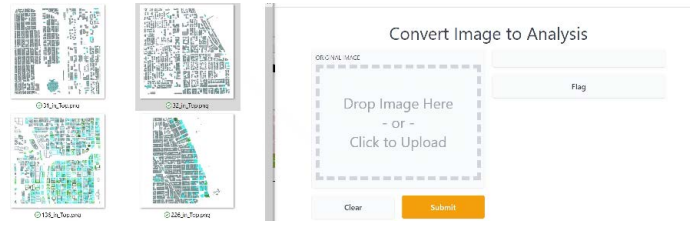
New cities Testing Set Data
(not seen by the algorythm before)

Input image | City AI predicted/ generated | Simulated/ expected



Temporary interface: Python platform Gradio

Video sample of the interface
Click on the image below to start the video showing the temporal interface in action.



Next Steps

A Creation of a permanent platform open to all the PW community.

A permanent site will be the most efficient deliverable to share this exercise and its capabilities with the whole PW community.

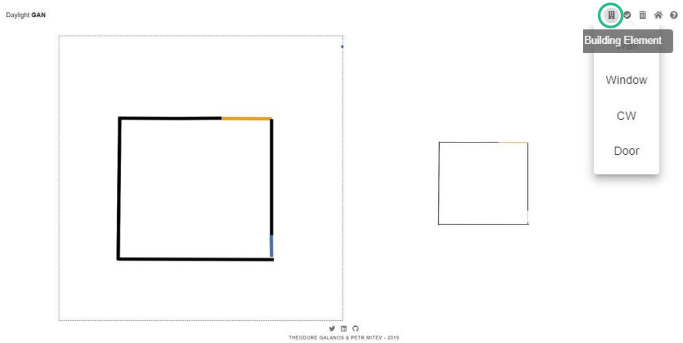
This platform could give access to early sustainability performance of urban areas with no need of specific skills (Rhino+Grasshopper) or training time. The platform could automatically apply the color range, given the heights information, and calculate the estimated Solar Access Simulation.

This real-time predictions could provide quick early insight into early design stages that could also be shared with clients while in meetings/workshops.

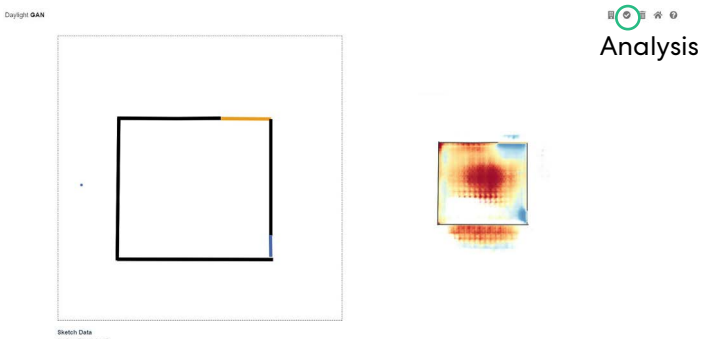
As reference of how a permanent platform could operate, images on the right show the Daylight GAN platform developed by T.Galanos and P.Mitev.

Daylight GAN: reference of permanent platform. LINK by Theodore Galanos and Petr Mitev.

First step. Drawing the floorplan elements: Wall, Window, Curtain Wall and Door. Each element is mapped with a different color.



Second step. The platform performs the daylight prediction analysis in real-time.



B Expand the analysis to Solar Radiation/ Pedestrian Wind Comfort analysis.

Based on the performance of the Pix2Pix GAN model on the prediction of Solar Access at urban scale, the team would like to expand this research into other environmental analysis and considerations such as:
- Solar Radiation Analysis: this would help teams have early insights into the renewable energy production capabilities of urban masterplans.

- Pedestrian Wind Comfort Analysis. Consideration of the pedestrian wind comfort analysis is critical when designing spaces that promote wellbeing and safe and pleasant use of public spaces.

Currently the Urban Design teams rely on external providers (SimScale in the case of London) for the assessment of pedestrian wind comfort analysis. The application of GANs to this exercise will make it more accessible, efficient and cost-effective for the PW teams to perform this urban analysis.

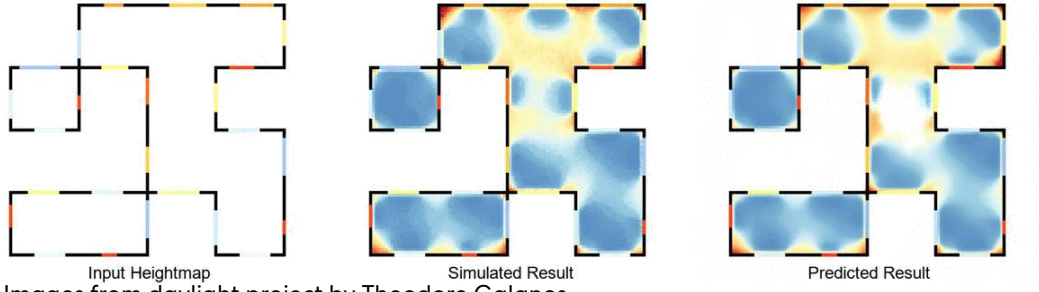
Previous research

The team has undertaken significant literature review covering topics related to this proposal such as the automation and fast-track of urban analysis or the application of AI (ML/GANs) within architecture and urban design research projects both for plans analysis and generation.

We list below some of the most interesting projects which in some cases include GitHub repositories that will be use for reference as departure points.

ML to speed up urban analysis;
URBANAI : project testing and developing a machine learning model to predict complex metrics at urban scale.

ML for daylight analysis;
DaylightGAN: project applying Cycle GAN for real-time daylight prediction at architecture scale (floorplans). Theodore Galanos



Images from daylight project by Theodore Galanos.
https://github.com/TheodoreGalanos/DaylightGAN

ML for architecture plans generation;
AI + Architecture: application of AI for plans analysis and generation, the generation is optimized based on performance of the plans against a metric framework.

ML for urban generation;
GANs for Urban Design: application of Generative Adversarial Networks for the design of an urban block considering various urban morphology styles. Data acquisition from Openstreetmap and Open Data portals of the cities.

This map does not exist: generation of OSM-style maps using StyleGAN2.

City AI relevance

We believe this project meets PW culture and goals and supports our core values towards a more sustainable, research-based and “Living Design”.

• **Sustainable agenda and climate action:** CityAI is a progression of the work that our UD team with the support of the research labs has been doing to deliver informed, assessed and sustainable urban projects to help our communities and cities make a change towards a climate agenda.

CityAI will push for even a greater role of environmental analysis on urban projects with quick assessments of large scales, given insight on district and city wide performance. It will also demonstrate that environmental consideration is part of our projects since inception and not an add-in at the final outcome.

• **Thought leaders and innovators:** the industry is advancing towards the implementation of AI into its design and delivery workflows. However, there is no commercial tool delivering environmental analysis for large urban scales as an automated and quick option. CityAI presents a new opportunity to present PW as thought leaders and innovators to our clients, collaborators and colleagues, and furthermore city leaders and private and public institutions working with the built environment and climate agendas.

• **Efficiency of workflows:** we are designers of the built environment and our value to the community relies on our propositions, strategic thinking and bold design. This is where our time should be dedicated to, let’s use AI to leverage our knowledge and assist us on creating sustainable and viable design solutions for our cities. CityAI will have the potential to speed-track relevant environmental analysis at larger urban scales, making assessment accessible and easy to integrate within workflows in an automated way.