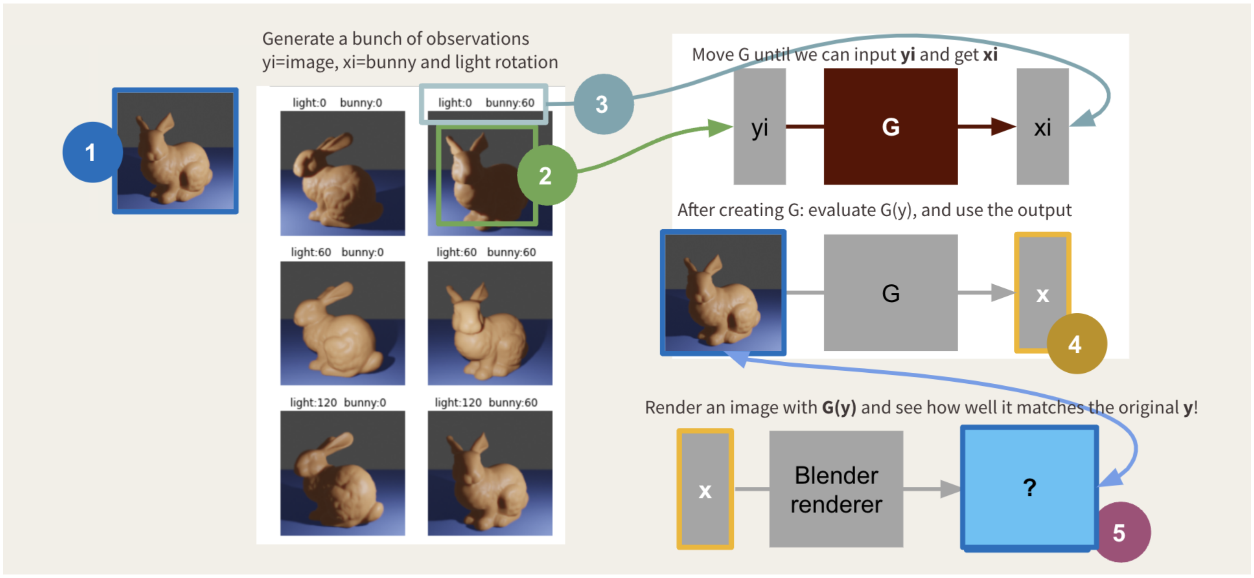
CS148 Homework 5 Part 2

Homework Due: Aug 4th **(Thursday)** at 11:59 PM PST
Quiz Date: Tuesday Aug 2nd

1 Assignment Outline

This is part 2 of HW5, which will only be comprised of the technical component.
In this homework, you will play around a bit with scripting in Blender, and look at how you can procedurally generate a few sets of "α, β, Render(bunny rotated by α, light rotated by β)", to create a function G that takes a render of this scene and predicts what (α, β) might be!
Note that you do not need any machine learning experience at all to do this homework, beyond the high level understanding of the "terrain traversal" analogy as given in lecture. The code to set up $\min_G ||G(y) - x||$ has been provided in the jupyter notebook.


1
Generate a bunch of observations
yi=image, xi=bunny and light rotation
light:0 bunny:0
light:0 bunny:60
light:60 bunny:0
light:60 bunny:60
light:120 bunny:0
light:120 bunny:60
2
3
Move G until we can input **yi** and get **xi**
yi
G
xi
After creating G: evaluate G(y), and use the output
G
x
4
Render an image with **G(y)** and see how well it matches the original **y**!
x
Blender renderer
?
5

You will work with a simple Blender scene setup that has a ground plane, a bunny, and an point light, and

1. Write a function that takes in the name of an object O in the scene and some angle θ as input, then sets the z-axis rotation of O to be θ degrees.

2. Write a function that loops through M evenly spaced z-axis rotation angles $\alpha_1, \ldots \alpha_M$ for the bunny and N evenly spaced z-axis rotation angles $\beta_1, \ldots \beta_N$ for the light, then render and save out the $M \times N$ images corresponding to all possible combinations (α_i, β_j).

3. Write out a dictionary from Blender, where the keys correspond to the output image names generated in the previous step, and the values correspond to α, β that was used to generate that image.

4. Run the existing code in the jupyter notebook, to generate a function $G(y)$ that takes in renders y and attempts to predict the bunny and lighting rotations (α, β).

5. Given some render y that wasn't part of the observations used to create G, take the predicted output $G(y)$ and render it in Blender.

These five items are labelled in the diagram above. Note again that you will not need to write any code for part 4.

Even though we are learning the scripting tools here in the context of a "Machine Learning toy example" (i.e. how you might generate data in Blender for use in a Machine Learning framework,) the Blender python API gives you a lot of computational control and access to parts of the scene, which can potentially improve and speedup your final project workflows if you decide to try to use it. For example, if you want to decide on the best position/angle for your main objects, you can consider using the scripting tools to sample positions/angles, and then render each combination out for side-by-side comparisons.

2 Quiz Questions

- Describe the similarities and differences between knowledge-based systems and machine learning, and describe one of the hybrid methods that we talked about in class (i.e. a method that incorporates a known mapping F and tries to learn a mapping G.) Describe specific details that would make us consider this particular example to be a hybrid approach.
- Describe what character rigging and character skinning is. What are the issues with linear blend skinning, and what sort of visual artefacts would we expect to see when using it?
- Advection is the process in which we move a quantity around in a grid of cells to update a field based on a velocity field. Explain the naive, intuitive approach to advecting a quantity in a grid cell and the issues that come with it. Then, explain how doing advection implicitly by tracing the grid cell "backwards in time" solves these issues.
- The end goal of simulating participating media is a density field that we maintain and update appropriately for ray tracing. Explain how we model light lost from absorption and out-scattering as we trace a light ray through a density field (You can pretend the coefficient of attenuation at a grid cell is exactly the fractional value stored in the density field if that helps you think about it). Then, explain at a high level how in-scattering adds extra light to regions in space (aka our grid) to cause smoke, clouds, etc to look like they have shape.

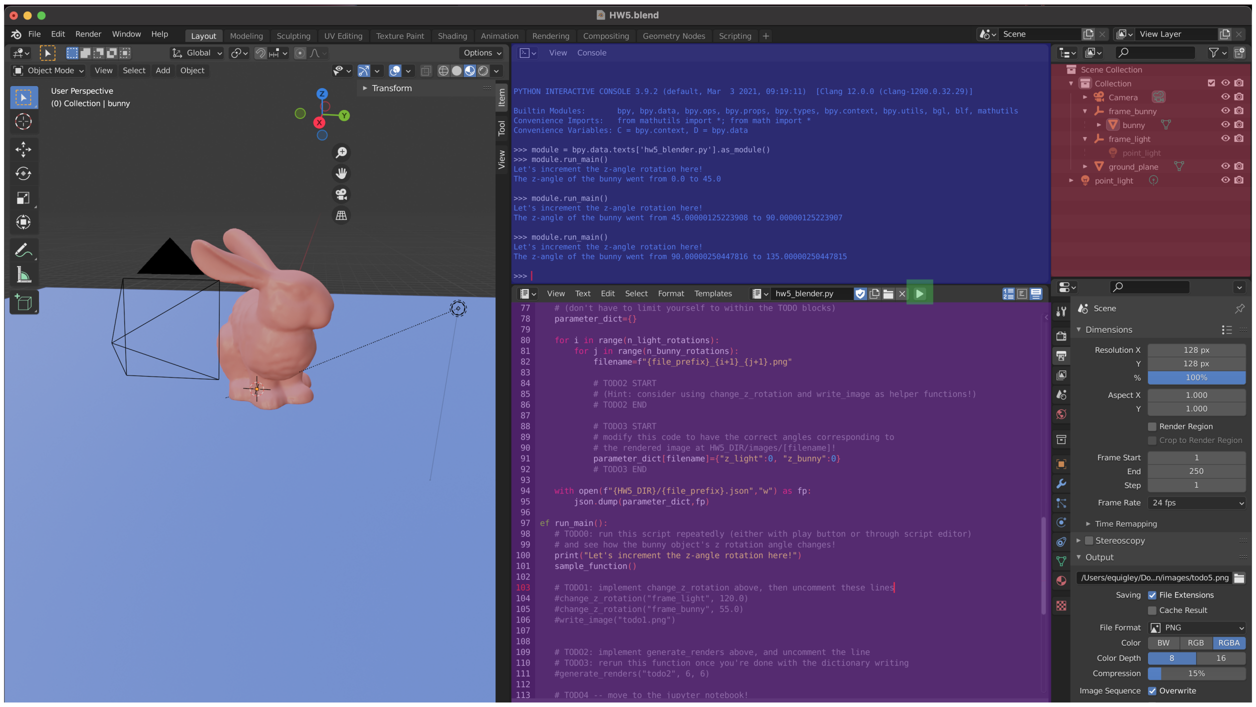
3 Setup

Please download `HW5_2.zip`, save it in some course directory `$CS148_DIR` on your machine, and unzip the file in this directory.
You will be doing all your coding in the Blender file `$CS148_DIR/HW5_2/HW5.blend` this week, whereas the only actions in `$CS148_DIR/HW5_2/HW5.ipynb` involve running each cell and responding to the short response questions.
Important Note: After you activate your anaconda environment and before you start the jupyter notebook, please do a one-time installation of the cpu-only version of pytorch via
```
conda install pytorch==1.10.1 torchvision==0.11.2 torchaudio==0.10.1 cpuonly -c pytorch
```

For those of you who have experience with pytorch, the example we'll run in step 4 of the assignment will not take more than 10 seconds on CPU. If you don't install via this exact command and run into machine/platform-specific GPU issues as a result, we will not be able to help!
When you are finished, print and save your Notebook as a PDF and submit to Gradescope.

4 General Interface

When you open up the blender file contained within the assignment directory, you should see something that looks like this:



```
PYTHON INTERACTIVE CONSOLE 3.9.2 (default, Mar  3 2021, 09:19:11)  [Clang 12.0.0 (clang-1200.0.32.29)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports:   from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> module = bpy.data.texts['hw5_blender.py'].as_module()
>>> module.run_main()
Let's increment the z-angle rotation here!
The z-angle of the bunny went from 0.0 to 45.0

>>> module.run_main()
Let's increment the z-angle rotation here!
The z-angle of the bunny went from 45.00000125223908 to 90.00000125223907

>>> module.run_main()
Let's increment the z-angle rotation here!
The z-angle of the bunny went from 90.00000250447816 to 135.00000250447815

>>> 
```

```
    # (don't have to limit yourself to within the TODO blocks)
    parameter_dict={}

    for i in range(n_light_rotations):
        for j in range(n_bunny_rotations):
            filename=f"{file_prefix}_{i+1}_{j+1}.png"

            # TODO2 START
            # (Hint: consider using change_z_rotation and write_image as helper functions!)
            # TODO2 END

            # TODO3 START
            # modify this code to have the correct angles corresponding to
            # the rendered image at HW5_DIR/images/[filename]!
            parameter_dict[filename]={"z_light":0, "z_bunny":0}
            # TODO3 END

    with open(f"{HW5_DIR}/{file_prefix}.json","w") as fp:
        json.dump(parameter_dict,fp)

ef run_main():
    # TODO0: run this script repeatedly (either with play button or through script editor)
    # and see how the bunny object's z rotation angle changes!
    print("Let's increment the z-angle rotation here!")
    sample_function()

    # TODO1: implement change_z_rotation above, then uncomment these lines
    #change_z_rotation("frame_light", 120.0)
    #change_z_rotation("frame_bunny", 55.0)
    #write_image("todo1.png")


    # TODO2: implement generate_renders above, and uncomment the line
    # TODO3: rerun this function once you're done with the dictionary writing
    #generate_renders("todo2", 6, 6)

    # TODO4 -- move to the jupyter notebook!
```

Scene
Dimensions
Resolution X 128 px
Y 128 px
% 100%
Aspect X 1.000
Y 1.000
☐ Render Region
☐ Crop to Render Region
Frame Start 1
End 250
Step 1
Frame Rate 24 fps
Time Remapping
☐ Stereoscopy
Output
/Users/equigley/Do...n/images/todo5.png
Saving ✓ File Extensions
☐ Cache Result
File Format PNG
Color BW RGB [✓ RGBA]
Color Depth [✓ 8] 16
Compression 15%
Image Sequence ✓ Overwrite

- The purple block displays Blender's internal text-editor, which is where the script you need to fill out (`hw5_blender.py`) is displayed.
- Clicking the play button (highlighted in green) will rerun the script and make changes to your scene (generally, Ctrl+Z will undo the changes). However, in cases where you need to debug, and wish to see the full error message, you may find using the in-built console (blue block on the middle top) useful. After each update to your script, you can call `module = bpy.data.texts['hw5_blender.py'].as_module()` in the console, followed by `module.run_main()` , to run the corresponding run_main() function defined in `hw5_blender.py` . For more workflow options, please see this link to the Blender python API.
- As you can see in the outliner (red block), the bunny and light are linked under "frame_bunny" and "frame_light" – utilizing parent transform frames like this give us more control over the parametrization of an object's motion. We didn't get to this demonstration in lecture, but note how we can now essentially sweep the point light in a circle around the bunny now with just a single z-axis rotation. If you tried to rotate the light directly in the light's own object space, the rotation would not cause any change in the light position at all (and rotating a point with zero volume / zero area similarly does not achieve anything either.)

As a result, in your code, you want to be manipulating these coordinate frames instead of the raw geometry/light directly.

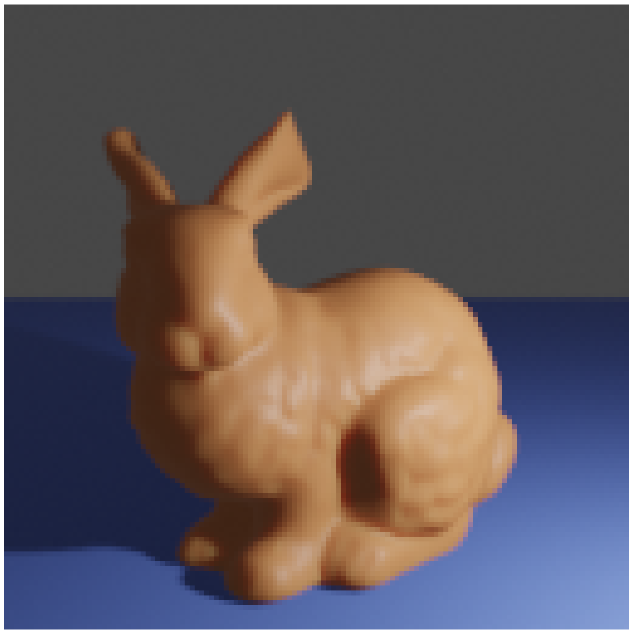
- Lastly, please do not edit anything in this Blender file besides the script. If you want to experiment some more, feel free to duplicate the file and do the testing in a new instance of this scene file!

5 Todos

We recommend following the comments in the python script directly. What you need to do is as follows:

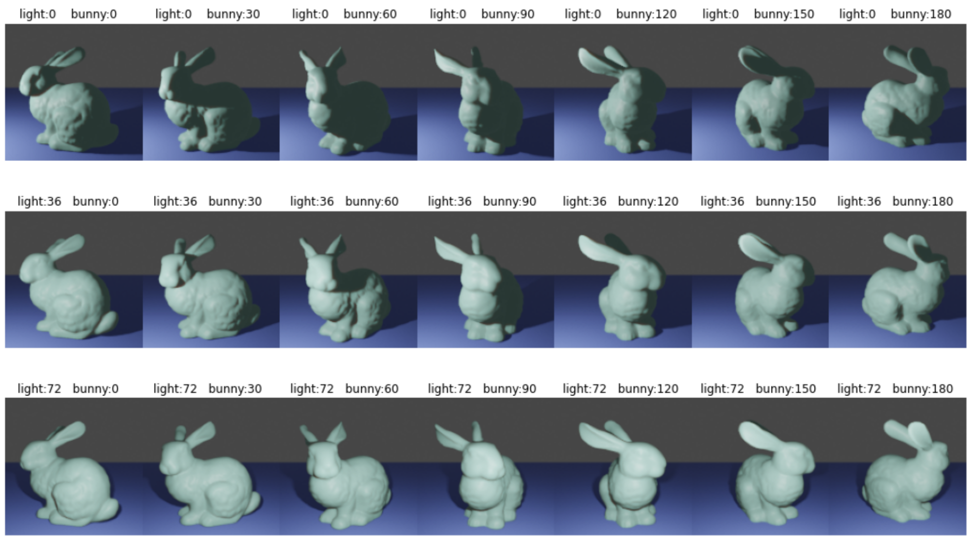
- TODO 0: Change the `HW5_DIR` as defined at the top of the script to point to your homework 5 directory. Trying running the script with the play button, and figure out how the script is setup.

- TODO 1: Write the helper function `change_z_rotation(object_name, z_angle_degrees)` to manipulate elements in the scene. Uncomment the relevant lines in run_main().
  Action: Check that `todo1.png` has been successfully rendered to `$HW5_DIR/images`. Run the corresponding cell in jupyter, and you should see a render similar to this one:

  The color of the bunny will be different, but the lighting and orientation of the bunny should match this reference image.
  Action: Fill out the short response answer by looking at the uncommented lines in Blender.

- TODO 2: Fill out the function `generate_renders()` to sample bunny and light rotations spaced evenly across [0,180] degrees, and generate a grid of images corresponding to the different combinations of bunny rotation and light rotation.
  Action: At this point, if you try to run the jupyter notebook, you should get a grid of images similar to this one but with the wrong labels:



- TODO 3: Fill out the function `generate_renders()` to populate and write out the parameter dictionary, where we store image filenames as keys and store the z angle rotations of both the bunny and the light as the values. Check the comments in the code for the detailed specifications.
  Action: At this point, if you try to run the jupyter notebook, you should get the grid of images from TODO2 but with correct labels. The grid example above is what you should be expecting for the first three rows of the grid (barring a different bunny color): in particular, each column corresponds to a fixed bunny orientation with varying lighting, and each row corresponds to fixed lighting with varying bunny orientation. The labels should reflect the rotation angle correctly.

- TODO 4: Once you've verified that the grid of images has corresponding labels that are correct, run the next cell in the jupyter notebook to fit the function G to this set of images and labels. G will try to produce the mapping that goes from images to labels (bunny+light rotations).
  After this cell finishes running, run the cell after it, which will take the image that you generated in TODO1, and pass it in as input to G. G will then try to estimate what the lighting and bunny orientations are from the image directly.
  Action: You do not need to do anything beyond making sure that the code runs without errors and prints out a prediction at the end. Note that you may get a warning about NNPACK if you are on a newer MacBook – please feel free to disregard.

- TODO 5: Take the prediction from TODO4, and render it in Blender by uncommenting and editing the relevant lines in the script editor.
  Action: Run the second to last cell in the notebook. You should see the original input image y, as well as the render produced by setting the bunny and light orientation to the values obtained by evaluating $G(y)$. The renders should look slightly different but not overly so.
  Action: Make sure to save the script in the script editor by going explicitly to Text→Save in the script editor panel (or Alt/Option+S. Note that Ctrl+S will not work.) Then run the very last cell in the jupyter notebook.

6 Resources

This assignment is meant to be a mini crash course to Blender scripting, which is a really nice and useful general tool for a big range of applications. Beyond computational approaches to scene design and data generation for machine learning applications, you can even use it to create custom UI panels that can be added to the Blender interface! To learn more, we'd recommend looking through the Blender API overview for things you may be interested in doing.