CS3242 3D Modeling and Animation

Lab 4: Particle Simulation

In this lab, we are going to explore how to simulate particle systems, first by using the particle simulation engine in Maya, and then by writing a Python script ourselves. If you have not yet noticed, this is our first lab about animation! ☺

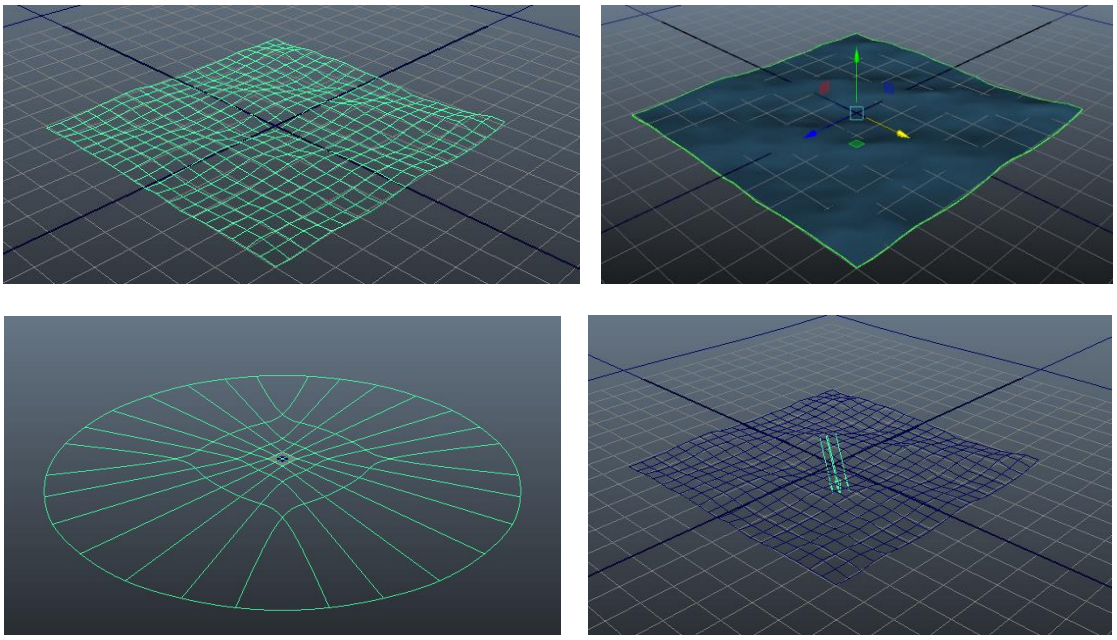
A. Ocean simulation

In this section, we will learn how to simulate an ocean in Maya and then render it.

1. Choose Dynamics menu set.

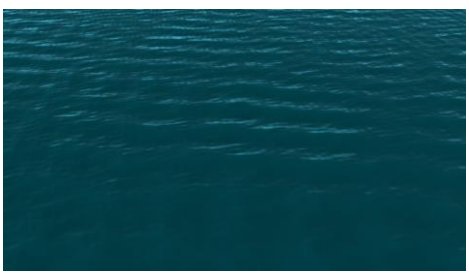
Select Fluid Effects > Ocean > Create Ocean.
Maya will automatically create an ocean model and a preview grid that displays a small part of the ocean surface.

Zoom out to see the entire ocean.
If the Smooth Shade All mode is on, you will see that the ocean material is set to a default blue color.

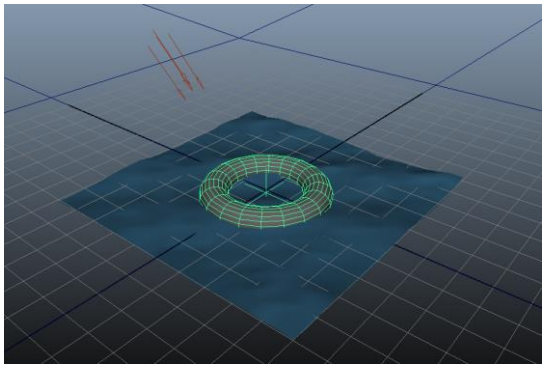


At this step, try move the slider in the timeline and see what happens.

2. Select Create > Lights > Directional Light. Change its intensity to 2.0, and rotate it towards the ocean. Render the current frame.



3. Maya is also capable of simulating objects that float on the ocean.



Create an object, e.g., a torus, and place it near to the ocean surface.

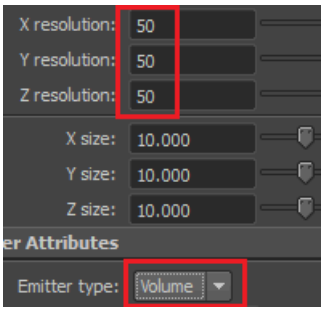
Select the cube and the ocean plane (the larger one, not the preview grid).
Navigate to Fluid Effects > Ocean > Make Boats.

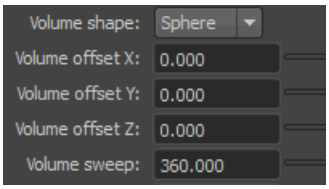
4. Move the slider in the timeline, or press play to play back the animation.

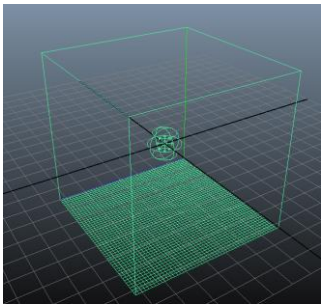
B. Particle simulation

As you have learnt in the lecture, particle system is a very powerful tool to simulate effects such as fire, smoke, or cloud. In Maya, we can create a basic particle system as follows.

1. For a particle system to work, we need to define a region in which the particle can move.
Choose Dynamics menu set and navigate to Fluid Effects > Create 3D Container with Emitter ❐.



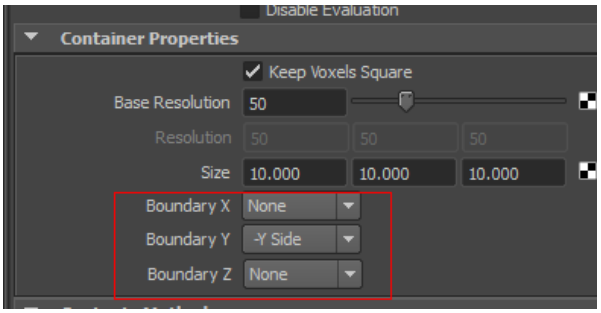


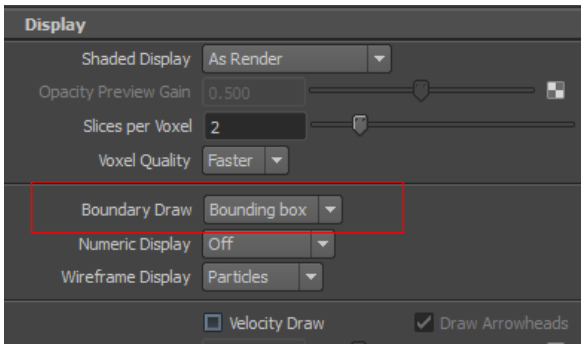


Set the resolution to 50, and the emitter type to Volume.
Set fluid dropoff to 0.2.

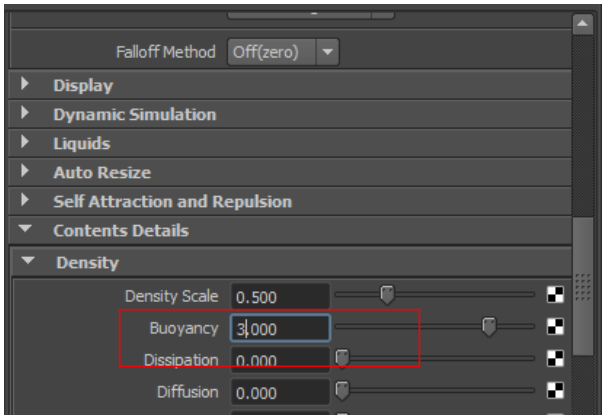
In the Volume Emitter Attributes section, set the volume shape to Sphere.
Click Apply. A particle container will be created in the scene.

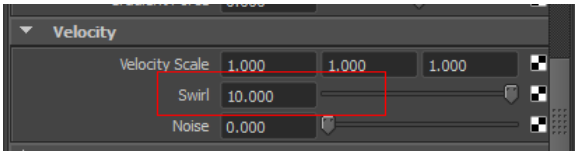
2. Select the container, and modify its properties as follows.

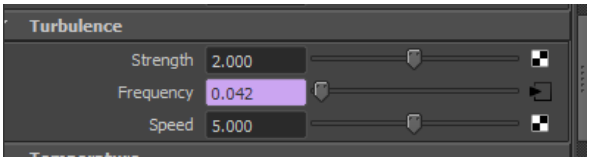




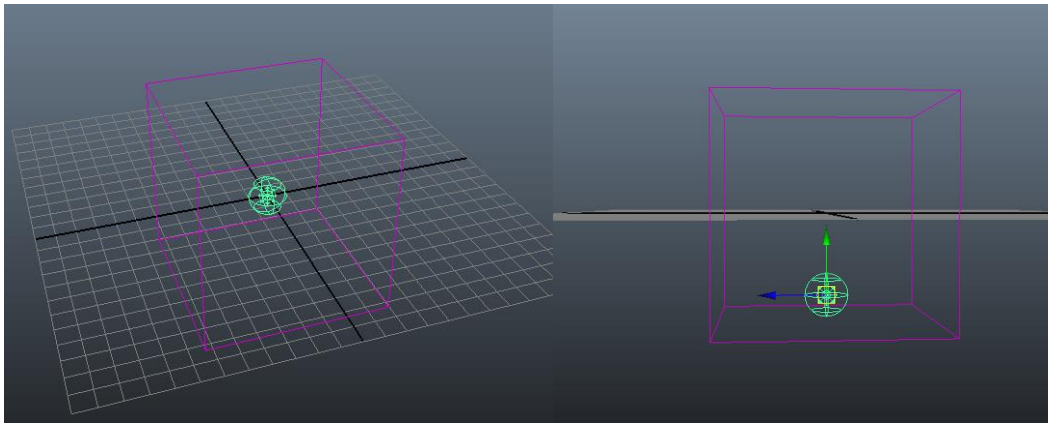
In the Content Details section, change the properties in the Density, Velocity, and Turbulence subsection as follows.







2. Select the emitter and move it to the bottom of the container:



3. Switch to smooth shade all, play the animation.

Try modifying the parameters to see how they affect the particle simulation. For example:

- Adjust viscosity.
- Set Buoyancy to a negative value.
- Adjust the density scale.

C. Particle simulation with Python scripting

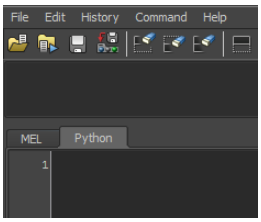
In this section, we are going to learn to code a simple Python script to make a particle move in Maya. We will simulate the trajectory of a particle in two examples:

1. The particle moves due to a circular motion field.
2. The particle is launched with an initial velocity and moves due to the gravity and viscous drag force.

We will implement the Euler method to find the location of the particle at a particular point of time. Our goal is to experiment with the Euler method and see how accurate it can simulate the motion of the particle.

Scripting in Maya

Here are the basic operations to work with the script editor in Maya:



1. Open the script editor by clicking the icon at the bottom right corner of the Maya user interface. Switch to the Python tab.
2. Go to File > Load Script and choose a file to open. The code in the file is now loaded into the code editor. (Make sure you do not load Python code into the MEL tab.)
3. To execute a few lines of the code, select them and press Ctrl + Enter.

If you press Ctrl + Enter without selecting the code, the code would be executed and then removed from the editor. To restore the code, press Ctrl + Z.

Particle representation

The particle attributes include a location, and a velocity vector. Each simulation step updates one or both of these attributes and save them for the next simulation step. In Maya, for convenience, we simply represent the particle by a small polygonal sphere. This allows us to use the default viewports, i.e., front view, or perspective view to display the particle, and the location and velocity of the particle can be stored as attributes of the polygonal sphere. We do not need to make any additional data structures.

A code template of each scenario is provided. Your task is to fill in the TODO sections in both of the examples and experiment with the parameters.
We assume the particle is the current selected object. To make it move, keep the object selected, and execute the script as the following instructions.

Scenario 1: Circular motion field

In this scenario, imagine the sun at the world origin, and the particle is the earth that rotates about the sun with a constant angular speed. At a particular location, the velocity vector is perpendicular to the line connecting the world origin and the particle center. Assume the velocity along the z-axis is always zero, and the angular speed is known.

In this example, try the below steps:

1. Create a small sphere in the scene and place it somewhere away from the origin.
2. Calculate the derivative `dxdt`, which is the velocity vector at the current location of the particle. Implement the `compute_circle_derivatives()` function.
   Hint: write the particle location in polar coordinates, and relate it with the angular velocity.
3. Implement the `euler()` function.
4. Select all the functions (leave out the last `maya_move()` line.)
   Press Ctrl + Enter to load all the functions into Python interpreter.
   Note: *every time you modify the functions, make sure they are reloaded to the interpreter*.
5. Select the sphere.
   Select the line `maya_move()` at the end of the code and execute it.

Each time `maya_move()` is called, the particle is expected to move a small step. A trajectory curve is also drawn in the scene to illustrate the path of the particle.

6. Try the simulation with different time steps.
   How does the trajectory look like? Are you able to make the trajectory circular?

Scenario 2: Launch a particle from the ground

In this example, assume there is a rock at the world origin. The rock is thrown into the air and after a while, it touches the ground. Assume the ground is the XZ plane. Let the initial velocity vector be `v0` (that has a positive Y value). The particle starts at time `t = 0` and touches the ground again after time `t = T`.

Assume that the particle is only driven by the gravity. (You can play with the viscous drag force after the verification if you want.)

Now let us simulate the movement of the particle, and examine if our simulation is accurate. Here are the steps:

Simulation

1. Create a new scene with a sphere.
2. Study the entire code from the previous scenario, and complete the TODO parts.
3. Load all functions into Python.
   Load the `const =` and `params =` lines near the end to load these variables to the interpreter. Again, remember to reload them if you change these values.
4. Select the sphere.
   Execute `maya_reset()` to move the particle back to the world origin.
5. Experiment with different time steps. Again, how does the trajectory look like?

Verification

6. Provide a formula to calculate the time T that the particle touches the ground again.
7. Execute `maya_create_destination()` to create a cube that denotes the expected location where the particle touches the ground. Note that this function assumes no viscous drag, and the initial Y velocity is positive.
   (You might need to zoom out to see the cube.)
8. Now execute `maya_move()` with different time steps and initial velocity.
   Would the particle be able to meet the expected target?

D. Further reading

1. PythonTutor, an online visualizer of Python code execution:
   http://www.pythontutor.com/

2. Python basics: http://www.learnpython.org/

3. nParticles: https://www.youtube.com/watch?v=F3K2x-Ahzkg

4. nCloth: http://www.youtube.com/watch?v=5MIRo8JBtDo
   Attach nCloth to Character: http://www.youtube.com/watch?v=asvSqkRGdlo

5. Deformers http://www.youtube.com/watch?v=tI2zSsKGmQM