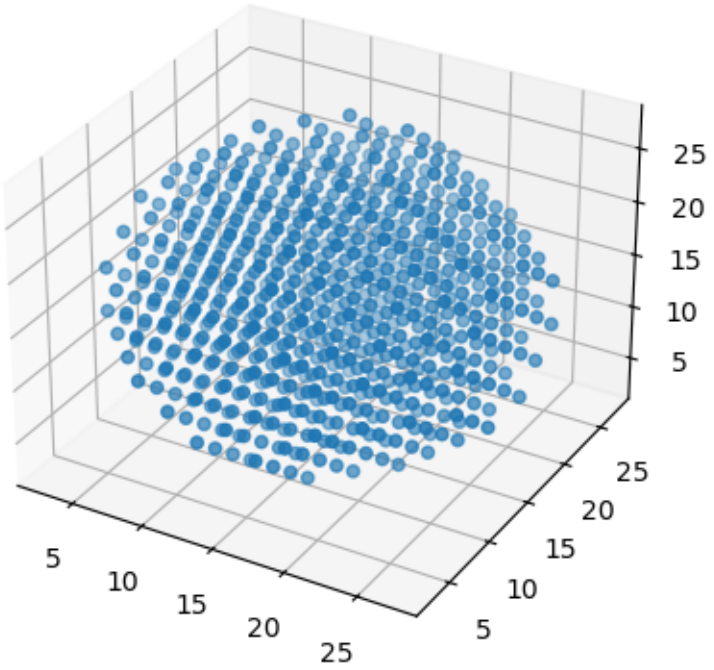
In [40]:
```
np_positions = nanoparticle.positions

print("First 10 atom positions: \n", np_positions[:10])

x = np_positions[:,0]
y = np_positions[:,1]
z = np_positions[:,2]

fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
ax.scatter(x, y, z)

plt.show()
```

```
First 10 atom positions:
 [[ 2.766  8.883 12.961]
 [ 4.805  6.844 12.961]
 [ 4.805  8.883 10.922]
 [ 2.766  8.883 17.039]
 [ 4.805  6.844 17.039]
 [ 4.805  8.883 15.   ]
 [ 4.805  8.883 19.078]
 [ 2.766 12.961  8.883]
 [ 4.805 10.922  8.883]
 [ 4.805 12.961  6.844]]
```

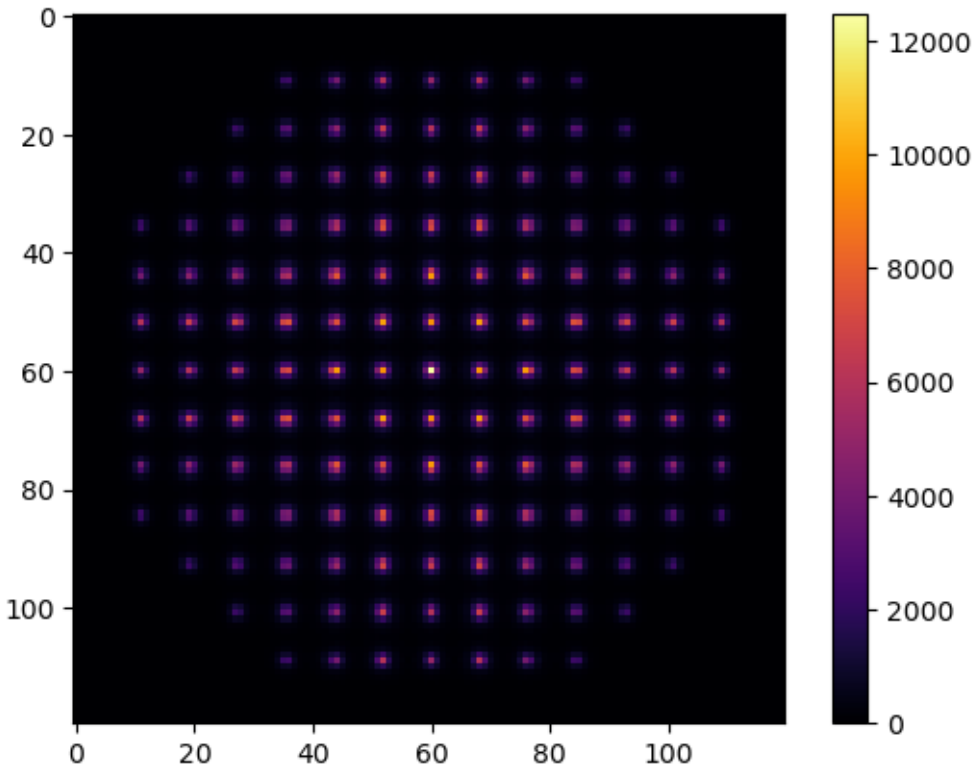


```
In [42]: grader.check("q1a")
```

```
Out[42]: q1a results: All test cases passed!
```

In [53]:
```
def ase_to_potential(np_sys):

    potential = abtem.Potential(atoms=np_sys, gpts=None, sampling=0.25, slice_thickness=0.25, p

    potentialArray = potential.build()

    return np.transpose(potentialArray.array.compute().sum(axis = 0))

test = ase_to_potential(nanoparticle)

plt.figure()
plt.imshow(test, cmap = "inferno")
plt.colorbar()
```

Out[53]: <matplotlib.colorbar.Colorbar at 0x16217e150>



In [55]:
```
grader.check("q1b")
```

Out[55]: q1b results: All test cases passed!

In [76]:
```
tilt_start = -90
tilt_end = 90

tilts_len = int((tilt_end - tilt_start)/5)
tilt_step = 5

steps = np.ones(tilts_len, dtype=int) *tilt_step
print(steps)
```

```
[5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]
```

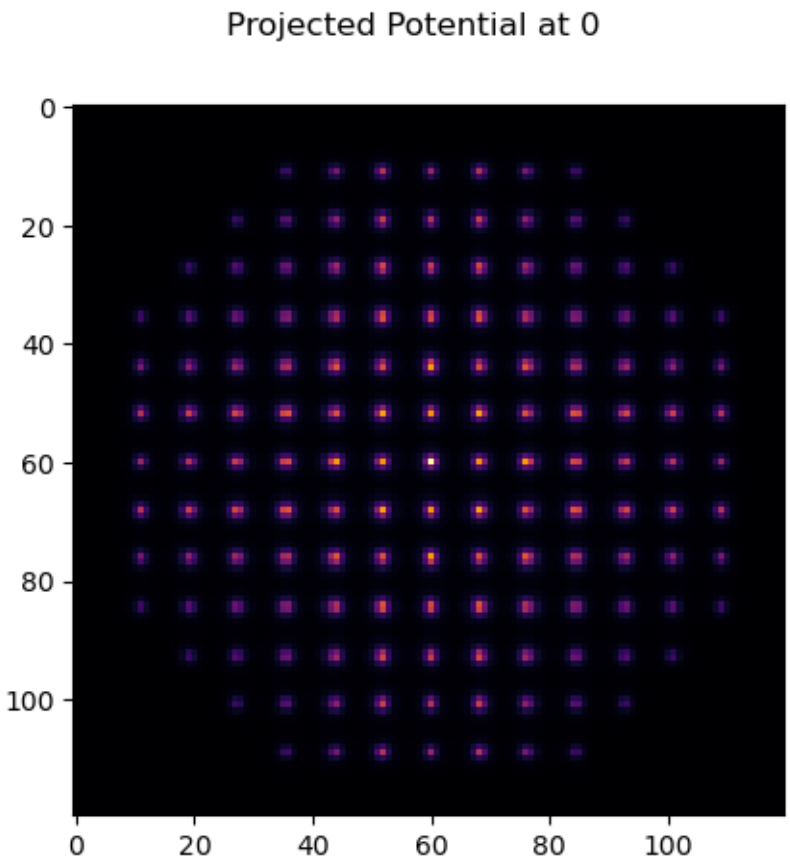
In [72]:
```
grader.check("q1c")
```

Out[72]:
```
q1c results: All test cases passed!
```

In [110]:
```
to_rotate = nanoparticle.copy() # Copies the original nanoparticle

center = (15, 15, 15) # Center of the unit cell.
to_rotate.rotate(tilt_start, v='y', center = center) # Rotate the nanoparticle to the startin

curr_angle = tilt_start # Current angle.
projected_images = [] # List for projected images.
tilt_angles = [] # List for current tilt angle.

# Append current starting orientation.
projected_images.append(ase_to_potential(to_rotate)) # Projected potential
tilt_angles.append(curr_angle) # Current tilt angle


for i in range(tilts_len):
    curr_angle = steps[i] + curr_angle
    tilt_angles.append(curr_angle)
    to_rotate.rotate(steps[i], v='y', center = center)
    projected_images.append(ase_to_potential(to_rotate))
```

In [132]:
```
'''
The below cell is not graded, but you should use ax.imshow, which scans through the projected
'''
# Check if your data makes sense. I.e, ind=0 and ind=-1 should be the same!
ind = 18
fig, ax = plt.subplots()
fig.suptitle(f"Projected Potential at {tilt_angles[ind]}")
im = projected_images[ind]

ax.imshow(im, cmap="inferno")
```

Out[132]:
```
<matplotlib.image.AxesImage at 0x1611c7320>
```



In [128]: grader.check("q1d")

Out[128]: q1d results: All test cases passed!