Meeting #8

10-27-21

Michael Lee

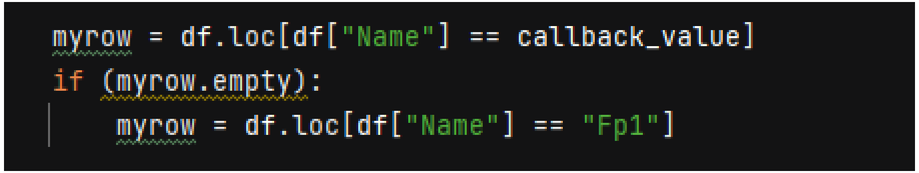
Deliverables

- Look into callback issue
- Display multiple electrode plot points

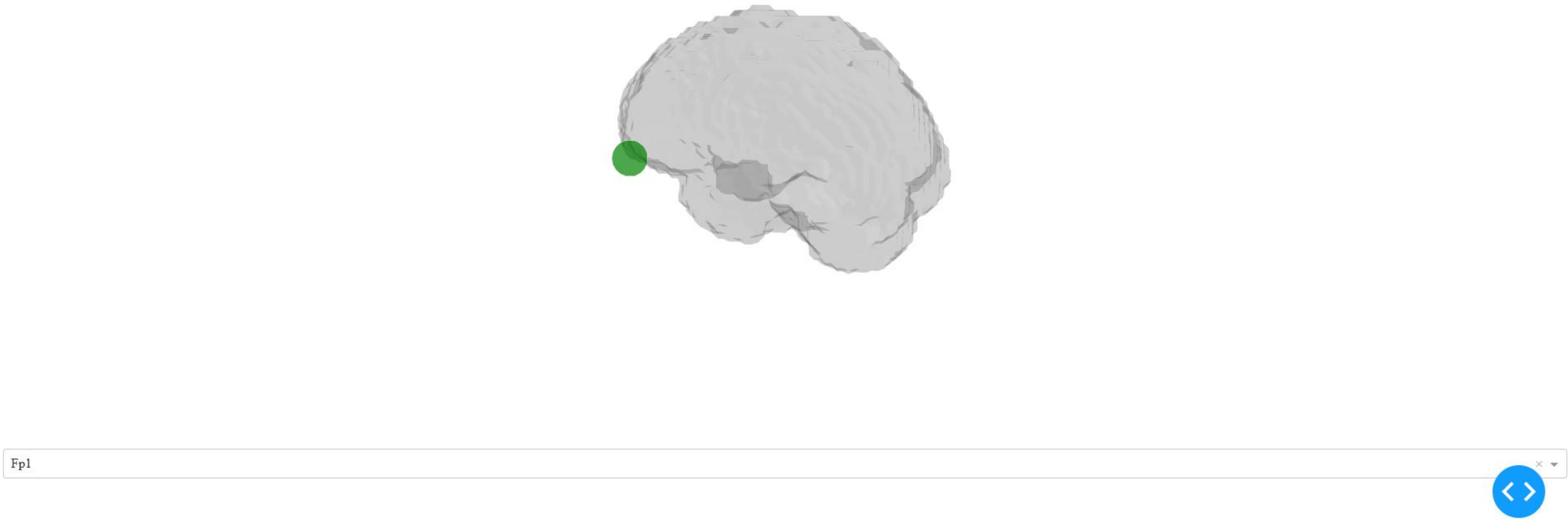
Methodology and Learnings

- Decoupling
- Callbacks

Results

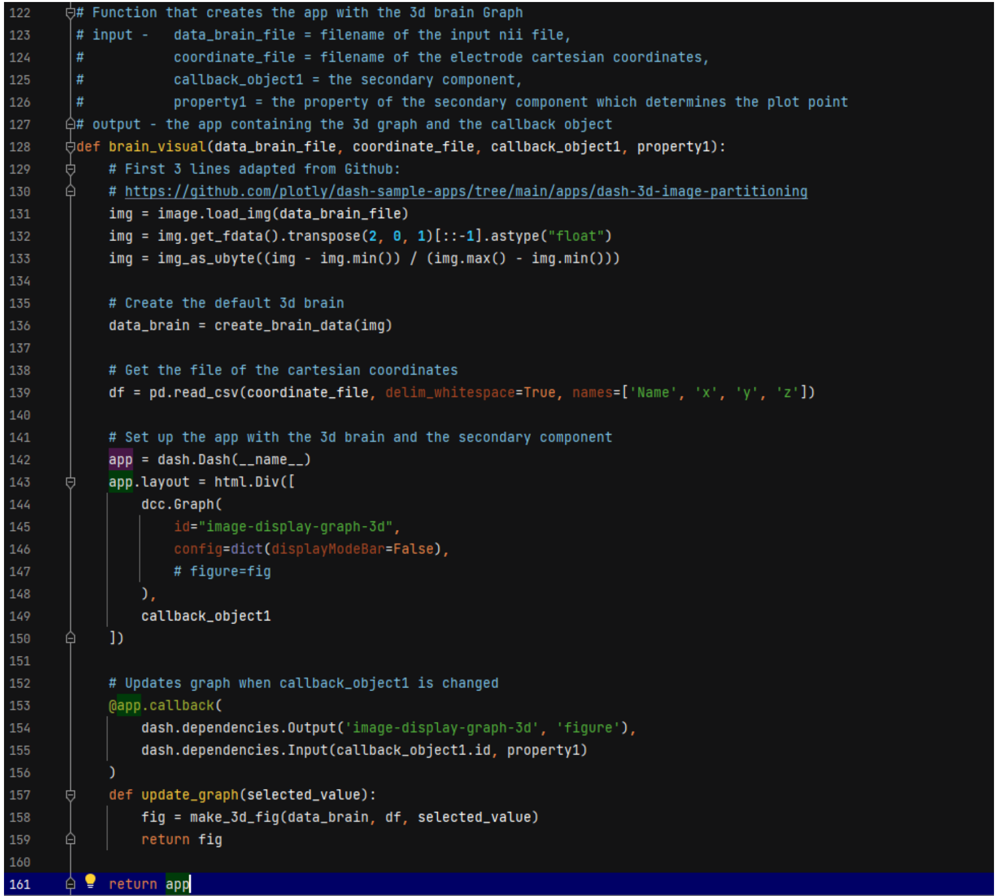


```
myrow = df.loc[df["Name"] == callback_value]
if (myrow.empty):
    myrow = df.loc[df["Name"] == "Fp1"]
```



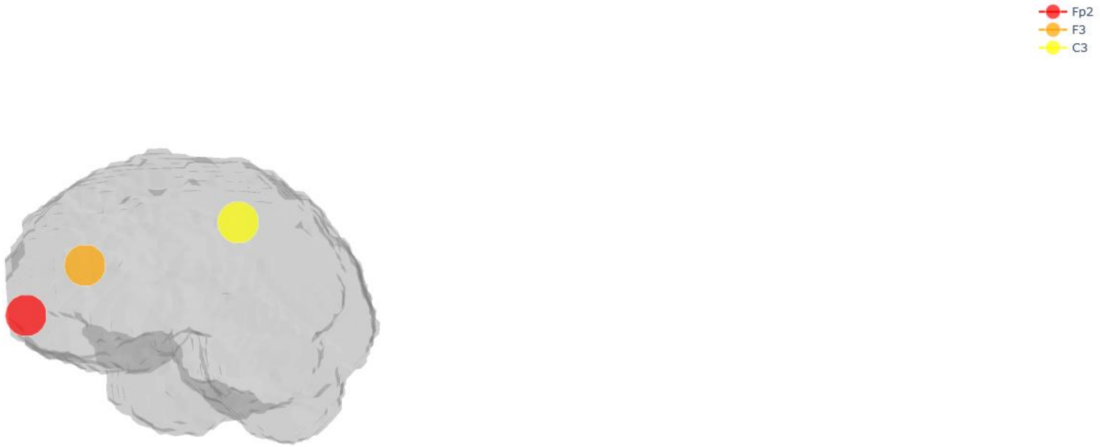
Fp1

Results



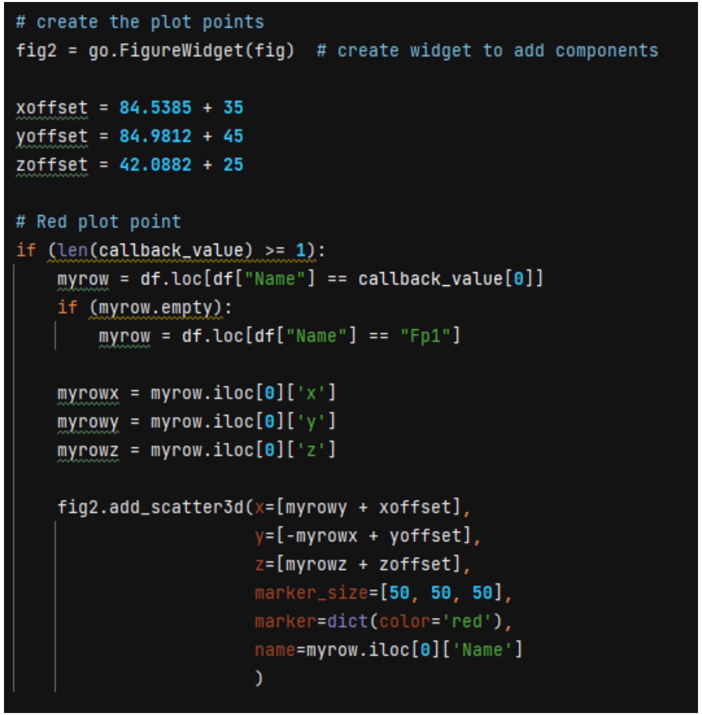
```python
# Function that creates the app with the 3d brain Graph
# input -   data_brain_file = filename of the input nii file,
#           coordinate_file = filename of the electrode cartesian coordinates,
#           callback_object1 = the secondary component,
#           property1 = the property of the secondary component which determines the plot point
# output - the app containing the 3d graph and the callback object
def brain_visual(data_brain_file, coordinate_file, callback_object1, property1):
    # First 3 lines adapted from Github:
    # https://github.com/plotly/dash-sample-apps/tree/main/apps/dash-3d-image-partitioning
    img = image.load_img(data_brain_file)
    img = img.get_fdata().transpose(2, 0, 1)[::-1].astype("float")
    img = img_as_ubyte((img - img.min()) / (img.max() - img.min()))

    # Create the default 3d brain
    data_brain = create_brain_data(img)

    # Get the file of the cartesian coordinates
    df = pd.read_csv(coordinate_file, delim_whitespace=True, names=['Name', 'x', 'y', 'z'])

    # Set up the app with the 3d brain and the secondary component
    app = dash.Dash(__name__)
    app.layout = html.Div([
        dcc.Graph(
            id="image-display-graph-3d",
            config=dict(displayModeBar=False),
            # figure=fig
        ),
        callback_object1
    ])

    # Updates graph when callback_object1 is changed
    @app.callback(
        dash.dependencies.Output('image-display-graph-3d', 'figure'),
        dash.dependencies.Input(callback_object1.id, property1)
    )
    def update_graph(selected_value):
        fig = make_3d_fig(data_brain, df, selected_value)
        return fig

    return app
```

Results

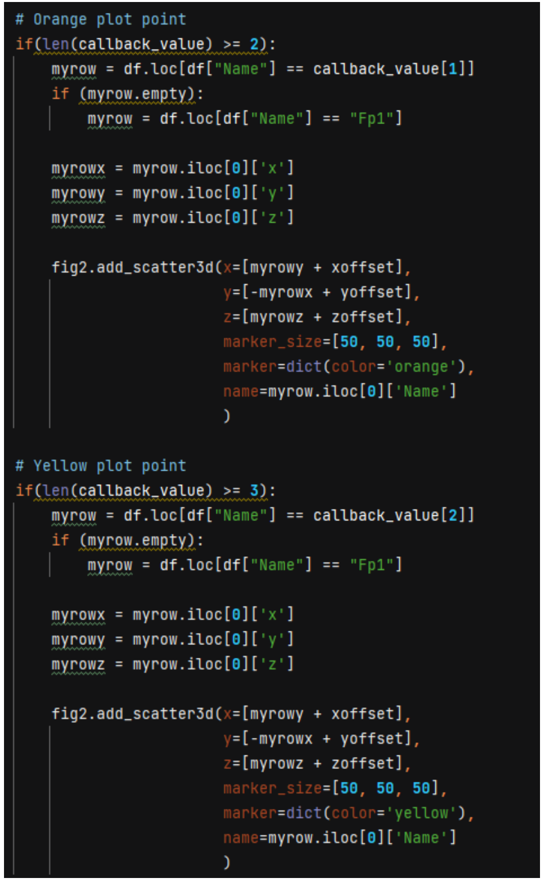


☐ Fp1 ☑ Fp2 ☑ F3 ☐ F4 ☑ C3

Results



```python
# create the plot points
fig2 = go.FigureWidget(fig)  # create widget to add components

xoffset = 84.5385 + 35
yoffset = 84.9812 + 45
zoffset = 42.0882 + 25

# Red plot point
if (len(callback_value) >= 1):
    myrow = df.loc[df["Name"] == callback_value[0]]
    if (myrow.empty):
        myrow = df.loc[df["Name"] == "Fp1"]

    myrowx = myrow.iloc[0]['x']
    myrowy = myrow.iloc[0]['y']
    myrowz = myrow.iloc[0]['z']

    fig2.add_scatter3d(x=[myrowy + xoffset],
                       y=[-myrowx + yoffset],
                       z=[myrowz + zoffset],
                       marker_size=[50, 50, 50],
                       marker=dict(color='red'),
                       name=myrow.iloc[0]['Name']
                       )
```



```python
# Orange plot point
if(len(callback_value) >= 2):
    myrow = df.loc[df["Name"] == callback_value[1]]
    if (myrow.empty):
        myrow = df.loc[df["Name"] == "Fp1"]

    myrowx = myrow.iloc[0]['x']
    myrowy = myrow.iloc[0]['y']
    myrowz = myrow.iloc[0]['z']

    fig2.add_scatter3d(x=[myrowy + xoffset],
                       y=[-myrowx + yoffset],
                       z=[myrowz + zoffset],
                       marker_size=[50, 50, 50],
                       marker=dict(color='orange'),
                       name=myrow.iloc[0]['Name']
                       )

# Yellow plot point
if(len(callback_value) >= 3):
    myrow = df.loc[df["Name"] == callback_value[2]]
    if (myrow.empty):
        myrow = df.loc[df["Name"] == "Fp1"]

    myrowx = myrow.iloc[0]['x']
    myrowy = myrow.iloc[0]['y']
    myrowz = myrow.iloc[0]['z']

    fig2.add_scatter3d(x=[myrowy + xoffset],
                       y=[-myrowx + yoffset],
                       z=[myrowz + zoffset],
                       marker_size=[50, 50, 50],
                       marker=dict(color='yellow'),
                       name=myrow.iloc[0]['Name']
                       )
```