Computer Graphics (UCS505)

3D EEG Signal Visualizer

B.E. 3rd Year – CSE

Submitted By –

Palak Aggarwal
102217184
Lavanya Panwar
102397006
Ashmeet Kaur
102217211

Submitted To – Dr. Amrita Kaur

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

INTRODUCTION

Project Overview:

3D EEG Signal Visualizer is a computer graphics project designed to visualize EEG (Electroencephalogram) data on a 3D brain model. The application maps electrode activity levels onto corresponding positions on the brain's surface, providing an intuitive way to understand neural activity distribution.

This tool is particularly useful for medical professionals and researchers who need to quickly identify areas of high and low brain activity based on EEG reports.

Key Features

- **3D Brain Model Rendering**: Loads and displays a 3D brain model using OpenGL.
- **Electrode Positioning**: Reads electrode coordinates from a JSON file and maps them onto the brain surface.
- **Activity Visualization**: Colors electrodes based on their activity levels (from another JSON file).
- **Color Mapping**: Blue → Green → Red gradient to represent low → medium → high activity.

Scope of the Project:

1. 3D Visualization of Brain Anatomy

This feature enables the real-time rendering of a high-quality, anatomically accurate 3D brain model using OpenGL. Users can freely rotate, zoom, and view the brain from any angle, which provides a realistic context for interpreting EEG data. The interactive 3D environment helps users understand spatial relationships between different brain regions, making it especially useful for educational and diagnostic purposes. The model serves as the canvas upon which all electrode and activity data are displayed, ensuring that the visualization

is not only scientifically accurate but also visually informative and user-friendly for those analyzing complex neurophysiological information.

2. Electrode Data Mapping

Electrode positions are imported from a structured JSON file containing 3D coordinates for each EEG sensor. The program reads this data and renders a visual representation of each electrode as a colored sphere placed accurately on the brain model. This mapping ensures that the EEG data corresponds to the correct anatomical regions of the brain. Users can also adjust the scale and offset of the electrode placements using keyboard inputs, enabling precise alignment in cases where coordinate normalization is required. This visual mapping plays a vital role in correlating the EEG readings with specific brain regions in an intuitive way.

3. EEG Activity Visualization

Activity values associated with each electrode are read from a separate JSON file and used to visually differentiate electrodes by their level of neural activity. Each electrode is assigned a color based on its activity intensity, allowing users to visually detect which brain areas are more or less active. The process updates dynamically when the program runs, offering a real-time representation of the patient's EEG data. This approach significantly enhances the readability of complex EEG datasets [1] and provides users with immediate, visual insights into neural patterns without the need to interpret numerical values manually or through statistical analysis.

4. Color Gradient for Activity Representation

To convey activity levels in an easily interpretable way, the application uses a smooth gradient color mapping: blue represents low activity, green indicates moderate activity, and red signifies high activity. This continuous gradient offers a visually intuitive cue for interpreting EEG data without numerical overlays. The color changes in real time as the data loads, making it possible to see subtle shifts in brain activity across the entire cortex. This visual coding helps users, especially those without deep technical backgrounds, grasp the significance of the data quickly and enhances the tool’s accessibility for both academic and

clinical applications.

5. User Interaction

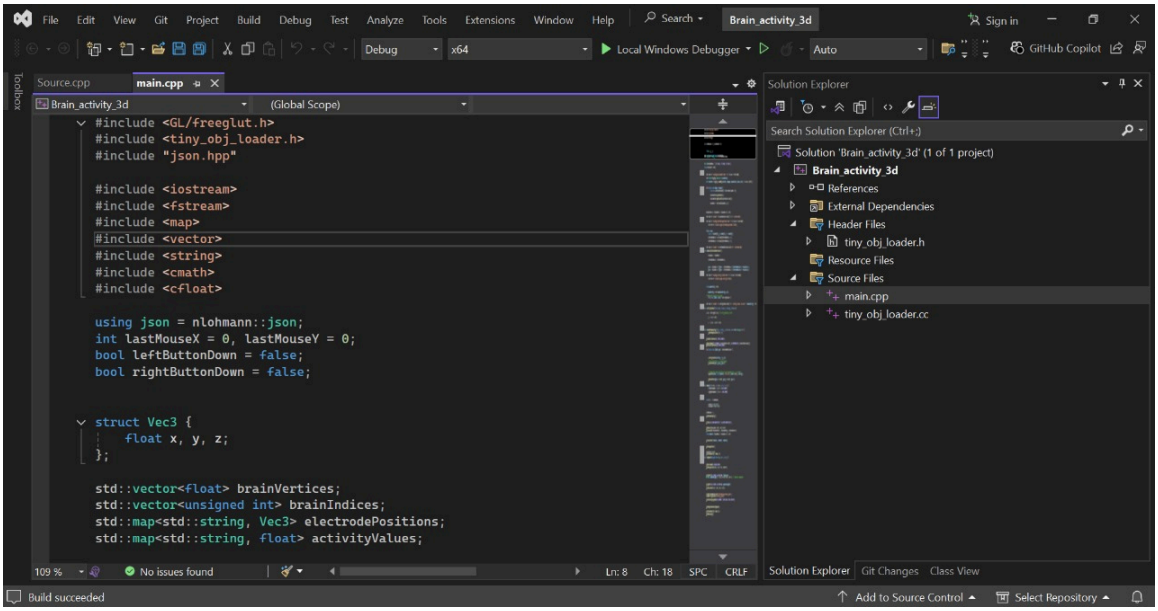
The application includes interactive features that enhance user control and improve the experience of exploring the data. Users can rotate and zoom the brain model using standard OpenGL navigation and adjust electrode positioning and scaling. These interactions allow for fine-tuning of the electrode visualization to ensure correct placement and better data interpretation. The rotation and zoom features enable detailed inspection of specific brain regions, which is critical when focusing on localized brain activity. Such interaction empowers users to tailor the visualization to their needs, improving both usability and analytical precision in real-world scenarios.

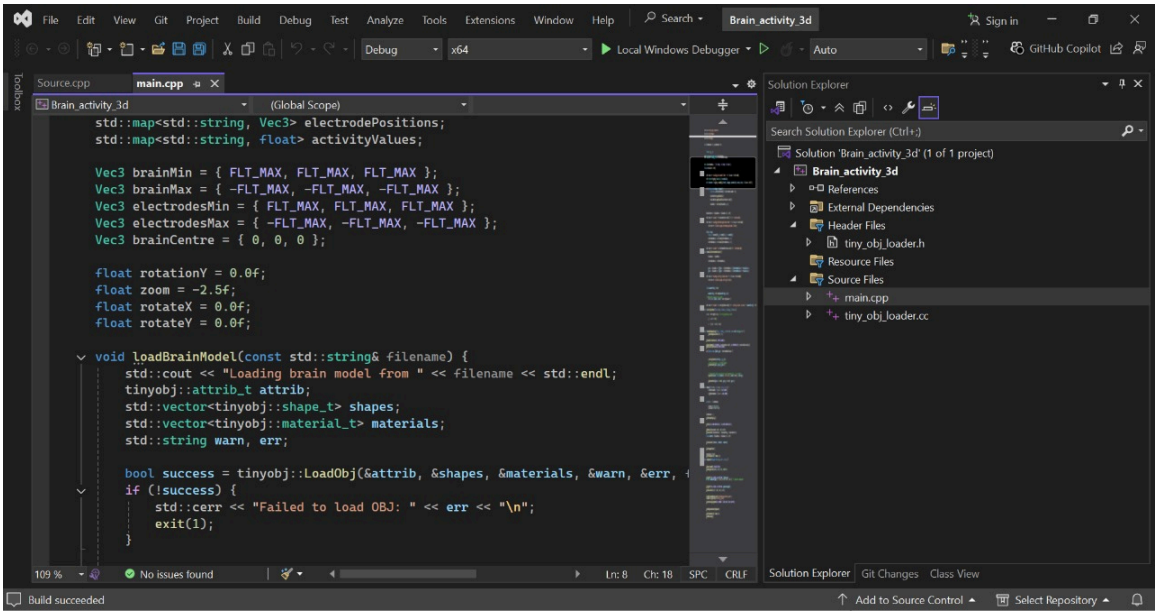
USER-DEFINED FUNCTIONS

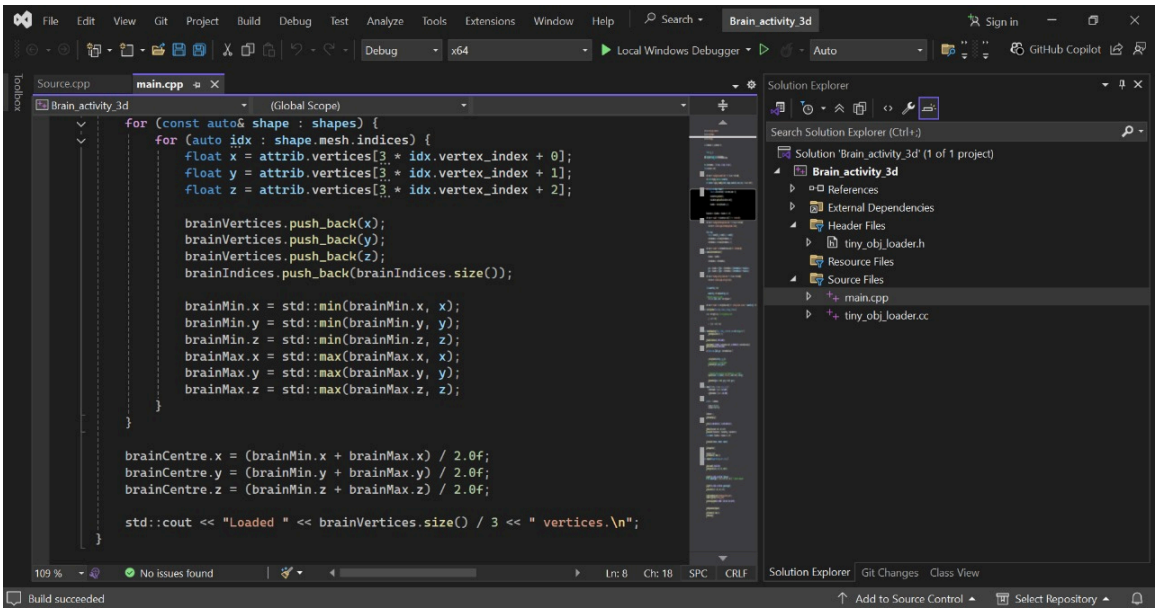
S No.	Function Name	Function Description
1	loadBrainModel(const std::string& filename)	Loads the brain model from an .obj file and fills the vertex and index buffers for rendering.
2	loadElectrodePositions(const std::string& filename)	Reads electrode positions from a JSON file and stores them in a label-to-coordinate map.
3	normalizeElectrodePositions()	Rescales electrode coordinates to match brain dimensions
4	loadActivityValues(const std::string& filename)	Loads electrode activity levels from a JSON file and stores them in a label-to-value map.
5	activityToColor(float value, float& r, float& g, float& b)	Converts a normalized activity value to an RGB color using a blue-green-red gradient.
6	renderBitmapString(float x, float y, void* font, const std::string& text)	Draws 2D text labels at specified screen positions

7	drawBrainModel()	Renders the brain model using the vertex array stored in memory.
8	drawElectrodes()	Draws electrodes as colored spheres based on activity values and mapped positions.
9	mouse(int button, int state, int x, int y)	Tracks mouse button presses for interaction
10	motion(int x, int y)	Handles mouse dragging for rotation/zoom
11	display()	Main rendering function that applies camera transforms and renders the brain and electrodes.
12	timer(int)	Updates the brain's Y-axis rotation and triggers re-rendering every 16ms.
13	keyboard(unsigned char key, int x, int y)	Handles key inputs to adjust the electrode scale, position, and to exit the app.
14	initOpenGL()	Initializes OpenGL settings like perspective, lighting, and depth testing.
15	main(int argc, char** argv)	Application entry point: loads files, initializes GLUT and OpenGL, and starts the main loop.

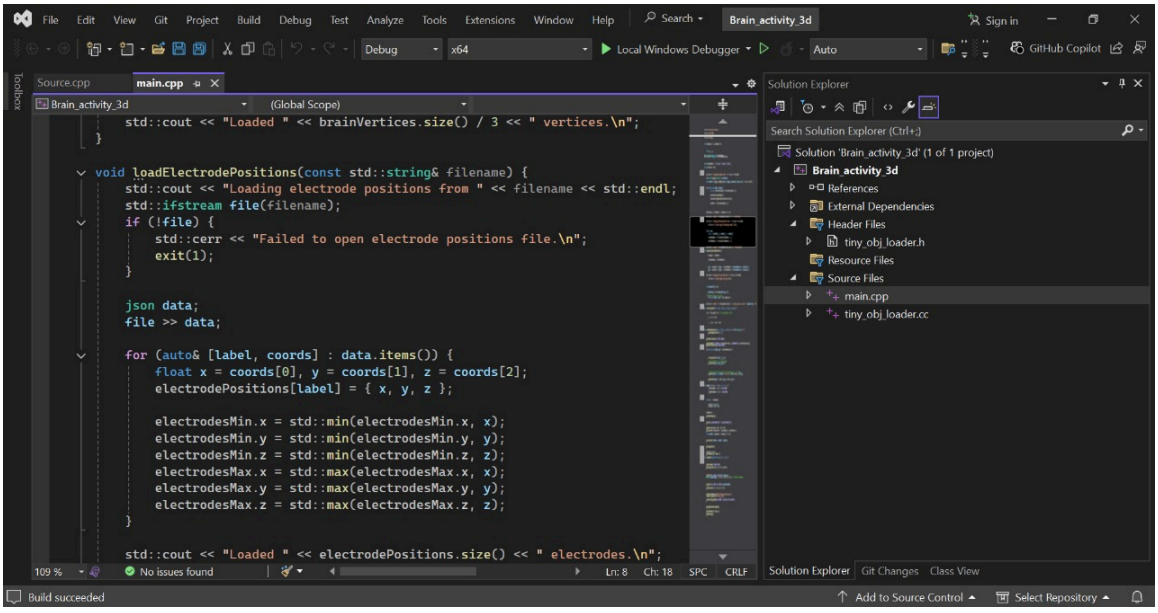
CODE SNIPPETS



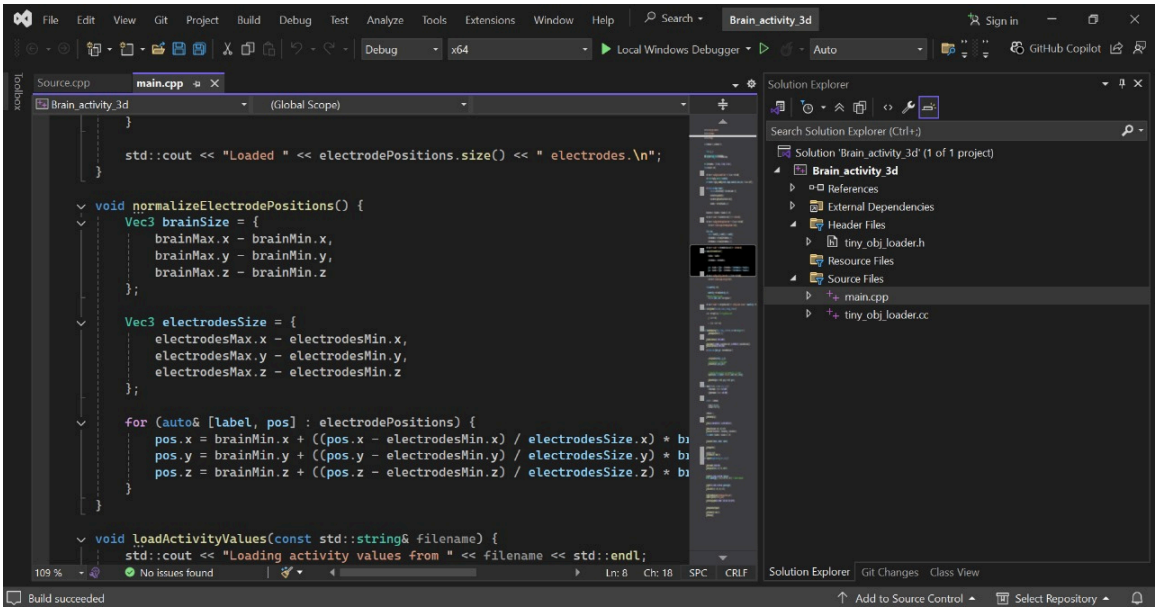




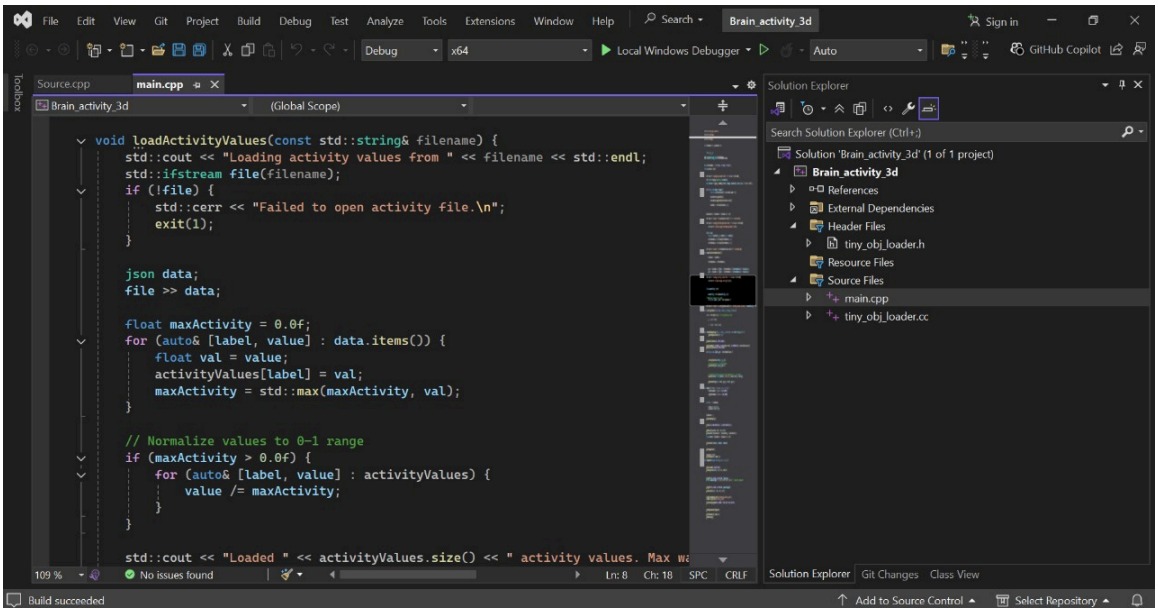
```
    for (const auto& shape : shapes) {
        for (auto idx : shape.mesh.indices) {
            float x = attrib.vertices[3 * idx.vertex_index + 0];
            float y = attrib.vertices[3 * idx.vertex_index + 1];
            float z = attrib.vertices[3 * idx.vertex_index + 2];

            brainVertices.push_back(x);
            brainVertices.push_back(y);
            brainVertices.push_back(z);
            brainIndices.push_back(brainIndices.size());

            brainMin.x = std::min(brainMin.x, x);
            brainMin.y = std::min(brainMin.y, y);
            brainMin.z = std::min(brainMin.z, z);
            brainMax.x = std::max(brainMax.x, x);
            brainMax.y = std::max(brainMax.y, y);
            brainMax.z = std::max(brainMax.z, z);
        }
    }

    brainCentre.x = (brainMin.x + brainMax.x) / 2.0f;
    brainCentre.y = (brainMin.y + brainMax.y) / 2.0f;
    brainCentre.z = (brainMin.z + brainMax.z) / 2.0f;

    std::cout << "Loaded " << brainVertices.size() / 3 << " vertices.\n";
}
```



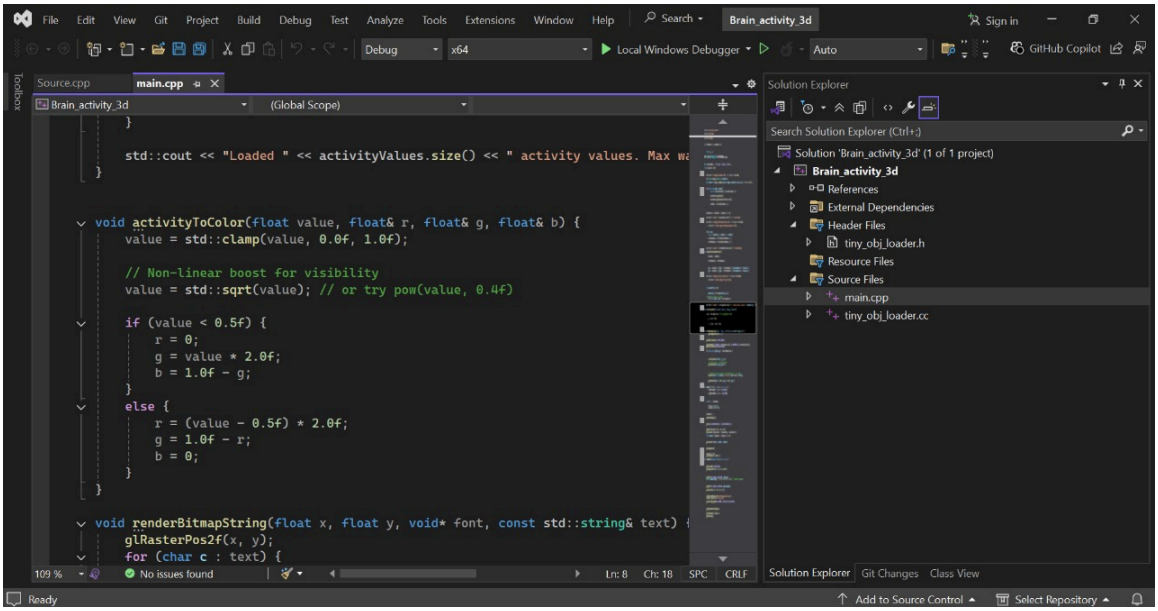
```
    std::cout << "Loaded " << brainVertices.size() / 3 << " vertices.\n";
}

void loadElectrodePositions(const std::string& filename) {
    std::cout << "Loading electrode positions from " << filename << std::endl;
    std::ifstream file(filename);
    if (!file) {
        std::cerr << "Failed to open electrode positions file.\n";
        exit(1);
    }

    json data;
    file >> data;

    for (auto& [label, coords] : data.items()) {
        float x = coords[0], y = coords[1], z = coords[2];
        electrodePositions[label] = { x, y, z };

        electrodesMin.x = std::min(electrodesMin.x, x);
        electrodesMin.y = std::min(electrodesMin.y, y);
        electrodesMin.z = std::min(electrodesMin.z, z);
        electrodesMax.x = std::max(electrodesMax.x, x);
        electrodesMax.y = std::max(electrodesMax.y, y);
        electrodesMax.z = std::max(electrodesMax.z, z);
    }

    std::cout << "Loaded " << electrodePositions.size() << " electrodes.\n";
```



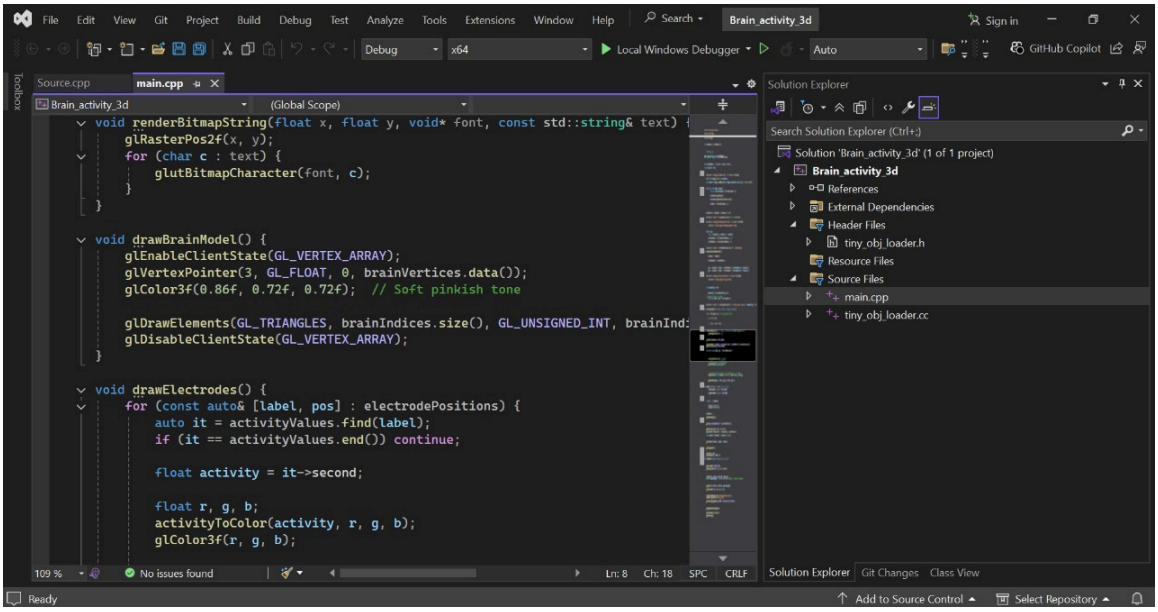
```
    }

    std::cout << "Loaded " << electrodePositions.size() << " electrodes.\n";
}

void normalizeElectrodePositions() {
    Vec3 brainSize = {
        brainMax.x - brainMin.x,
        brainMax.y - brainMin.y,
        brainMax.z - brainMin.z
    };

    Vec3 electrodesSize = {
        electrodesMax.x - electrodesMin.x,
        electrodesMax.y - electrodesMin.y,
        electrodesMax.z - electrodesMin.z
    };

    for (auto& [label, pos] : electrodePositions) {
        pos.x = brainMin.x + ((pos.x - electrodesMin.x) / electrodesSize.x) * br
        pos.y = brainMin.y + ((pos.y - electrodesMin.y) / electrodesSize.y) * br
        pos.z = brainMin.z + ((pos.z - electrodesMin.z) / electrodesSize.z) * br
    }
}

void loadActivityValues(const std::string& filename) {
    std::cout << "Loading activity values from " << filename << std::endl;
```



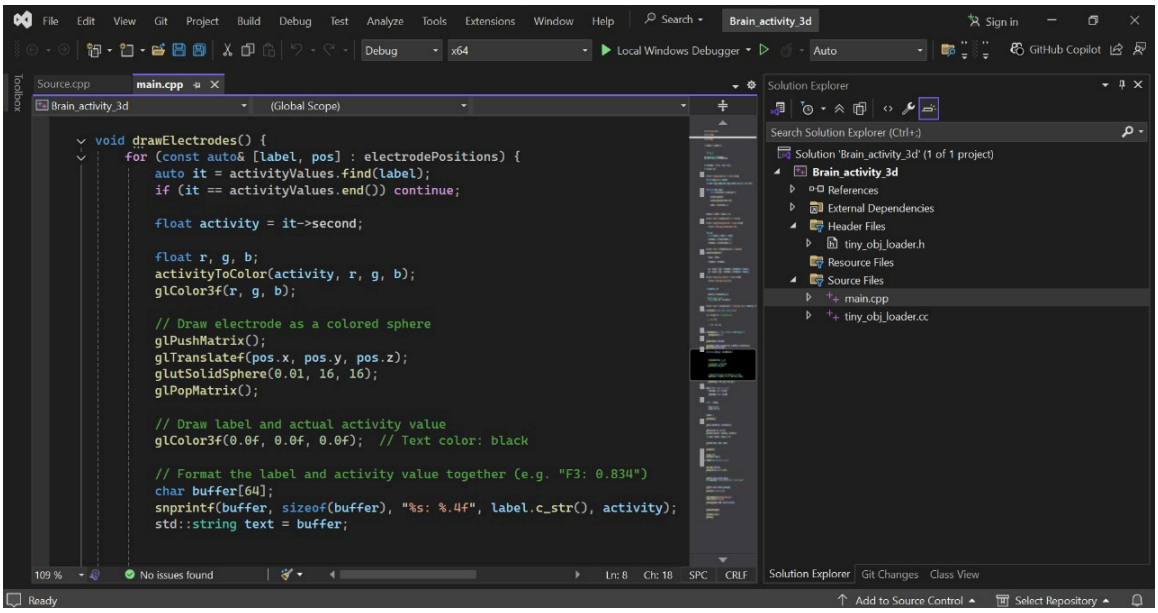
```
void loadActivityValues(const std::string& filename) {
    std::cout << "Loading activity values from " << filename << std::endl;
    std::ifstream file(filename);
    if (!file) {
        std::cerr << "Failed to open activity file.\n";
        exit(1);
    }

    json data;
    file >> data;

    float maxActivity = 0.0f;
    for (auto& [label, value] : data.items()) {
        float val = value;
        activityValues[label] = val;
        maxActivity = std::max(maxActivity, val);
    }

    // Normalize values to 0-1 range
    if (maxActivity > 0.0f) {
        for (auto& [label, value] : activityValues) {
            value /= maxActivity;
        }
    }

    std::cout << "Loaded " << activityValues.size() << " activity values. Max wa
```

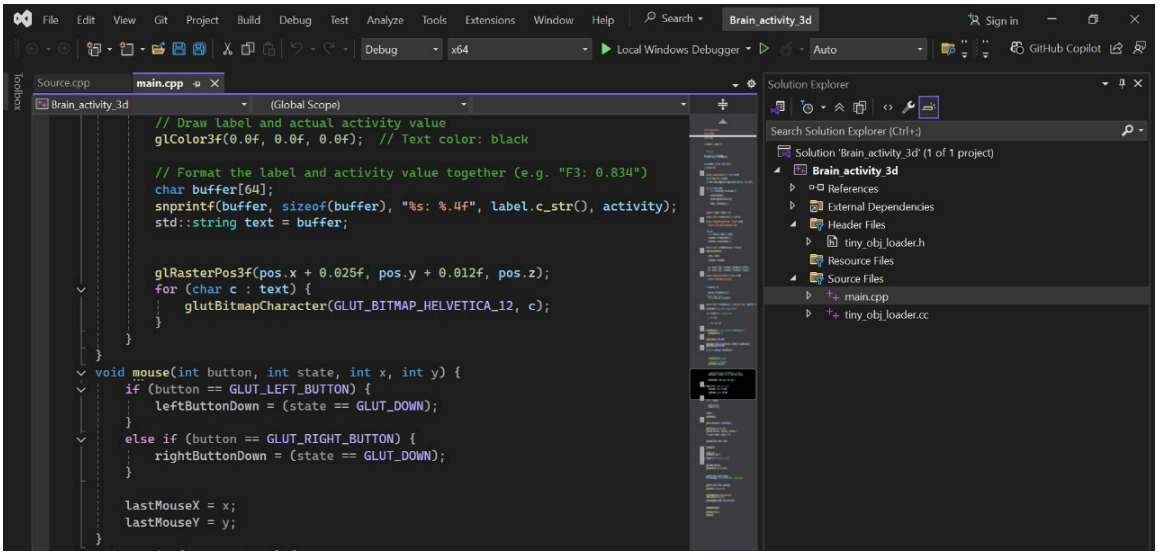


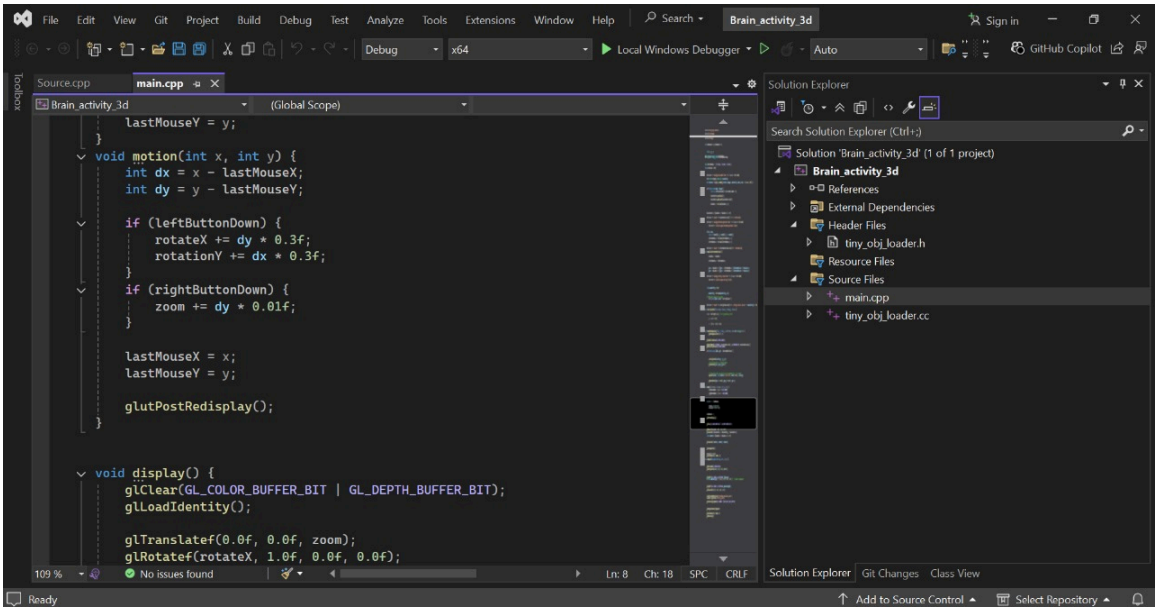
```
    }

    std::cout << "Loaded " << activityValues.size() << " activity values. Max wa
}


void activityToColor(float value, float& r, float& g, float& b) {
    value = std::clamp(value, 0.0f, 1.0f);

    // Non-linear boost for visibility
    value = std::sqrt(value); // or try pow(value, 0.4f)

    if (value < 0.5f) {
        r = 0;
        g = value * 2.0f;
        b = 1.0f - g;
    }
    else {
        r = (value - 0.5f) * 2.0f;
        g = 1.0f - r;
        b = 0;
    }
}

void renderBitmapString(float x, float y, void* font, const std::string& text) {
    glRasterPos2f(x, y);
    for (char c : text) {
```

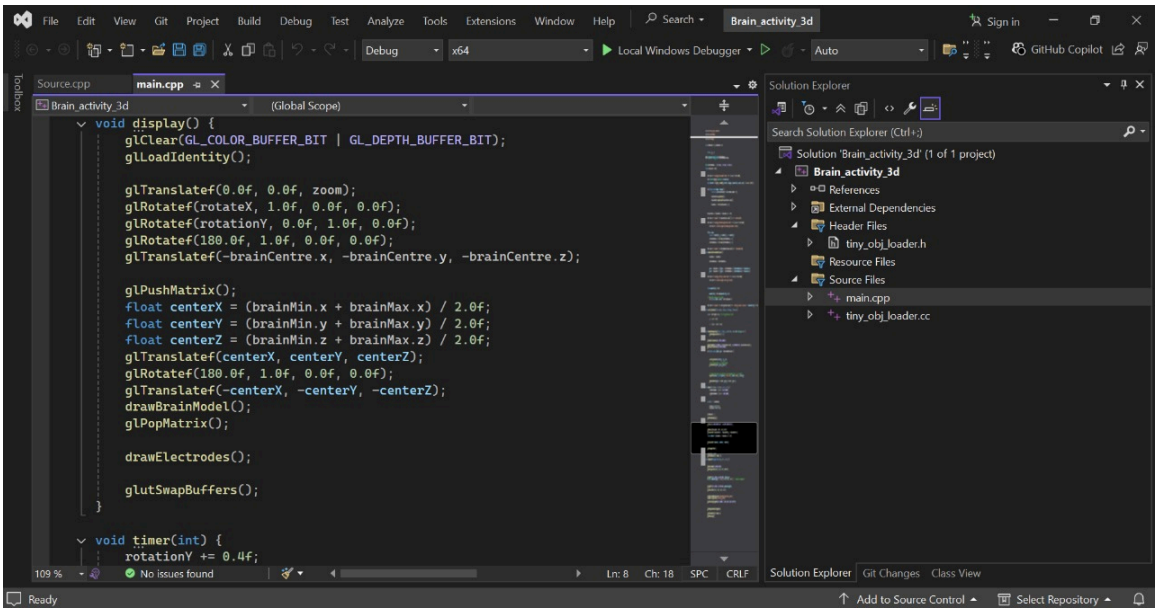


```
void renderBitmapString(float x, float y, void* font, const std::string& text) {
    glRasterPos2f(x, y);
    for (char c : text) {
        glutBitmapCharacter(font, c);
    }
}

void drawBrainModel() {
    glEnableClientState(GL_VERTEX_ARRAY);
    glVertexPointer(3, GL_FLOAT, 0, brainVertices.data());
    glColor3f(0.86f, 0.72f, 0.72f);  // Soft pinkish tone

    glDrawElements(GL_TRIANGLES, brainIndices.size(), GL_UNSIGNED_INT, brainInd
    glDisableClientState(GL_VERTEX_ARRAY);
}

void drawElectrodes() {
    for (const auto& [label, pos] : electrodePositions) {
        auto it = activityValues.find(label);
        if (it == activityValues.end()) continue;

        float activity = it->second;

        float r, g, b;
        activityToColor(activity, r, g, b);
        glColor3f(r, g, b);
```

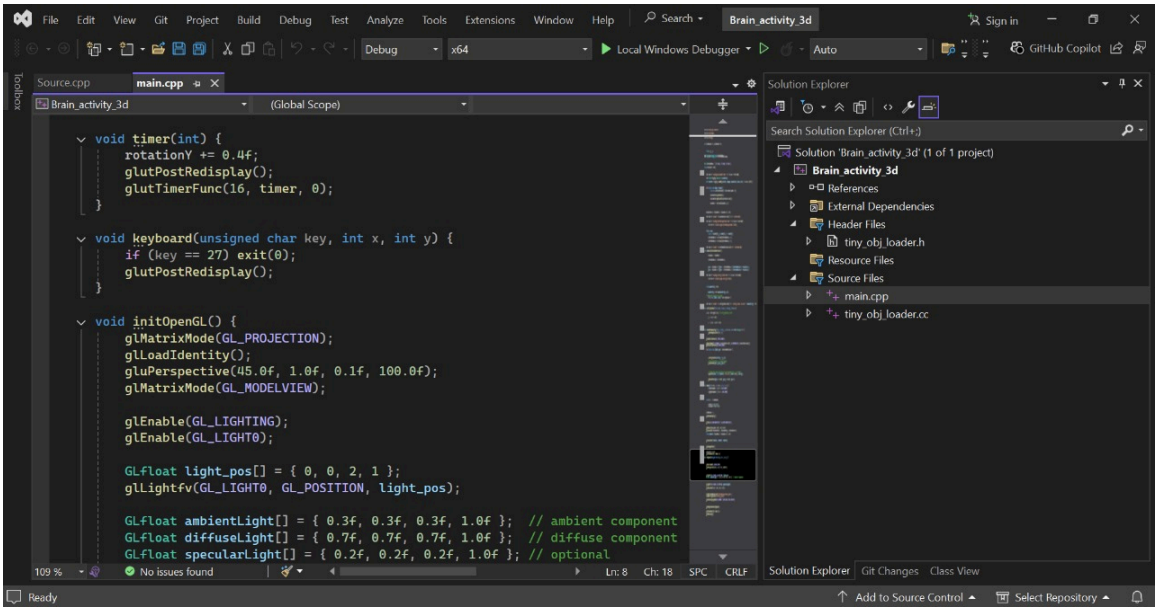

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
Brain_activity_3d
Sign in
Debug
x64
Local Windows Debugger
Auto
GitHub Copilot
Source.cpp
main.cpp
Brain_activity_3d
(Global Scope)
void drawElectrodes() {
for (const auto& [label, pos] : electrodePositions) {
auto it = activityValues.find(label);
if (it == activityValues.end()) continue;
float activity = it->second;
float r, g, b;
activityToColor(activity, r, g, b);
glColor3f(r, g, b);
// Draw electrode as a colored sphere
glPushMatrix();
glTranslatef(pos.x, pos.y, pos.z);
glutSolidSphere(0.01, 16, 16);
glPopMatrix();
// Draw label and actual activity value
glColor3f(0.0f, 0.0f, 0.0f); // Text color: black
// Format the label and activity value together (e.g. "F3: 0.834")
char buffer[64];
snprintf(buffer, sizeof(buffer), "%s: %.4f", label.c_str(), activity);
std::string text = buffer;
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Brain_activity_3d' (1 of 1 project)
Brain_activity_3d
References
External Dependencies
Header Files
tiny_obj_loader.h
Resource Files
Source Files
main.cpp
tiny_obj_loader.cc
109 %
No issues found
Ln: 8
Ch: 18
SPC
CRLF
Solution Explorer
Git Changes
Class View
Ready
Add to Source Control
Select Repository


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
Brain_activity_3d
Sign in
Debug
x64
Local Windows Debugger
Auto
GitHub Copilot
Source.cpp
main.cpp
Brain_activity_3d
(Global Scope)
// Draw label and actual activity value
glColor3f(0.0f, 0.0f, 0.0f); // Text color: black
// Format the label and activity value together (e.g. "F3: 0.834")
char buffer[64];
snprintf(buffer, sizeof(buffer), "%s: %.4f", label.c_str(), activity);
std::string text = buffer;
glRasterPos3f(pos.x + 0.025f, pos.y + 0.012f, pos.z);
for (char c : text) {
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, c);
}
}
}
void mouse(int button, int state, int x, int y) {
if (button == GLUT_LEFT_BUTTON) {
leftButtonDown = (state == GLUT_DOWN);
}
else if (button == GLUT_RIGHT_BUTTON) {
rightButtonDown = (state == GLUT_DOWN);
}
lastMouseX = x;
lastMouseY = y;
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Brain_activity_3d' (1 of 1 project)
Brain_activity_3d
References
External Dependencies
Header Files
tiny_obj_loader.h
Resource Files
Source Files
main.cpp
tiny_obj_loader.cc

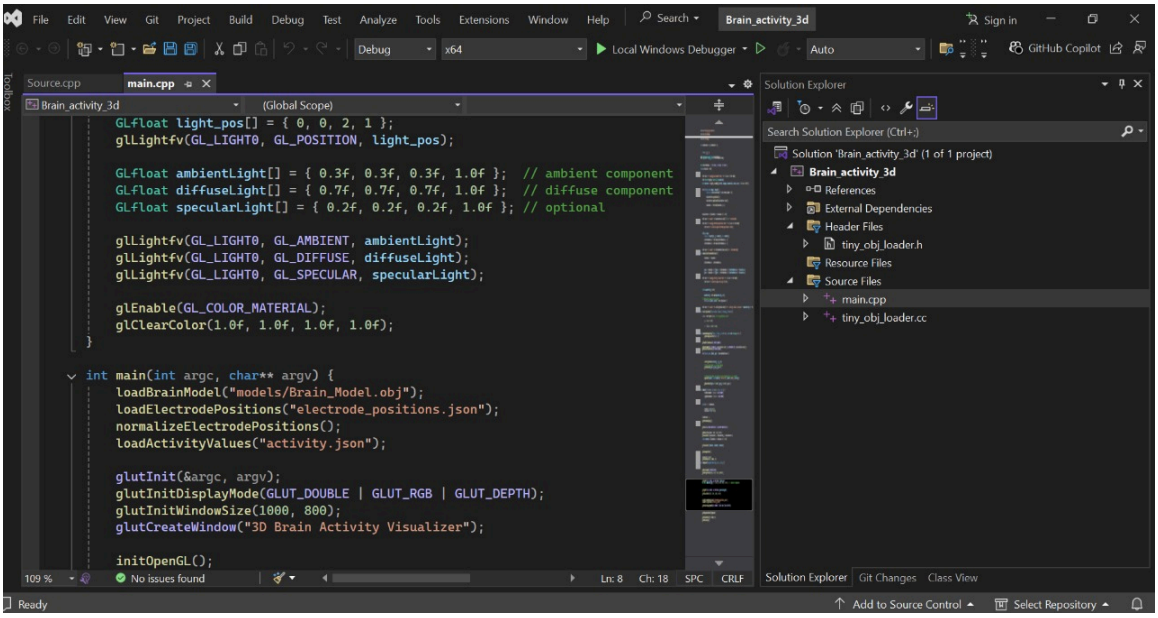

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
Brain_activity_3d
Sign in
Debug
x64
Local Windows Debugger
Auto
GitHub Copilot
Source.cpp
main.cpp
Brain_activity_3d
(Global Scope)
lastMouseY = y;
}
void motion(int x, int y) {
int dx = x - lastMouseX;
int dy = y - lastMouseY;
if (leftButtonDown) {
rotateX += dy * 0.3f;
rotationY += dx * 0.3f;
}
if (rightButtonDown) {
zoom += dy * 0.01f;
}
lastMouseX = x;
lastMouseY = y;
glutPostRedisplay();
}
void display() {
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glTranslatef(0.0f, 0.0f, zoom);
glRotatef(rotateX, 1.0f, 0.0f, 0.0f);
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Brain_activity_3d' (1 of 1 project)
Brain_activity_3d
References
External Dependencies
Header Files
tiny_obj_loader.h
Resource Files
Source Files
main.cpp
tiny_obj_loader.cc
109 %
No issues found
Ln: 8
Ch: 18
SPC
CRLF
Solution Explorer
Git Changes
Class View
Ready
Add to Source Control
Select Repository



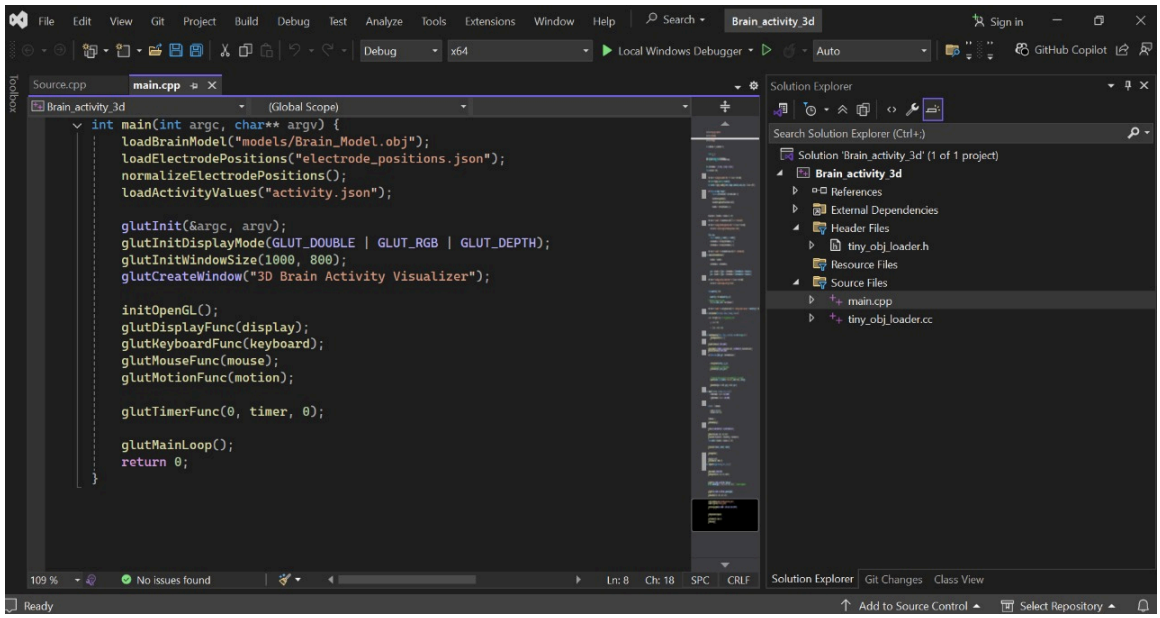
```
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glTranslatef(0.0f, 0.0f, zoom);
    glRotatef(rotateX, 1.0f, 0.0f, 0.0f);
    glRotatef(rotationY, 0.0f, 1.0f, 0.0f);
    glRotatef(180.0f, 1.0f, 0.0f, 0.0f);
    glTranslatef(-brainCentre.x, -brainCentre.y, -brainCentre.z);

    glPushMatrix();
    float centerX = (brainMin.x + brainMax.x) / 2.0f;
    float centerY = (brainMin.y + brainMax.y) / 2.0f;
    float centerZ = (brainMin.z + brainMax.z) / 2.0f;
    glTranslatef(centerX, centerY, centerZ);
    glRotatef(180.0f, 1.0f, 0.0f, 0.0f);
    glTranslatef(-centerX, -centerY, -centerZ);
    drawBrainModel();
    glPopMatrix();

    drawElectrodes();

    glutSwapBuffers();
}

void timer(int) {
    rotationY += 0.4f;
```



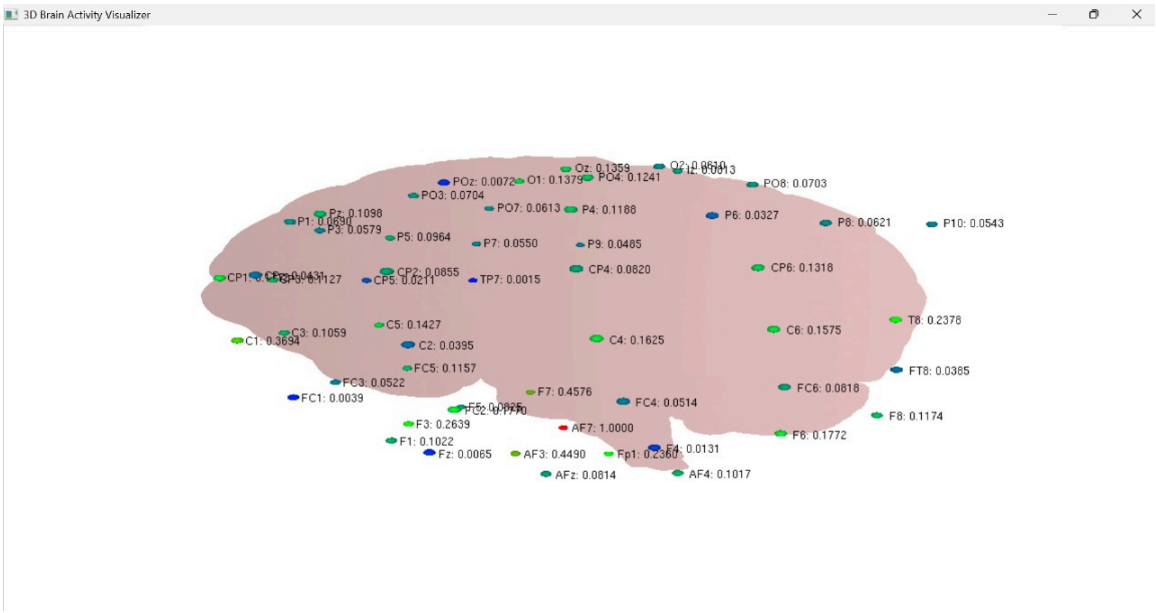
```
void timer(int) {
    rotationY += 0.4f;
    glutPostRedisplay();
    glutTimerFunc(16, timer, 0);
}

void keyboard(unsigned char key, int x, int y) {
    if (key == 27) exit(0);
    glutPostRedisplay();
}

void initOpenGL() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, 1.0f, 0.1f, 100.0f);
    glMatrixMode(GL_MODELVIEW);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    GLfloat light_pos[] = { 0, 0, 2, 1 };
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos);

    GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };  // ambient component
    GLfloat diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };  // diffuse component
    GLfloat specularLight[] = { 0.2f, 0.2f, 0.2f, 1.0f }; // optional
```

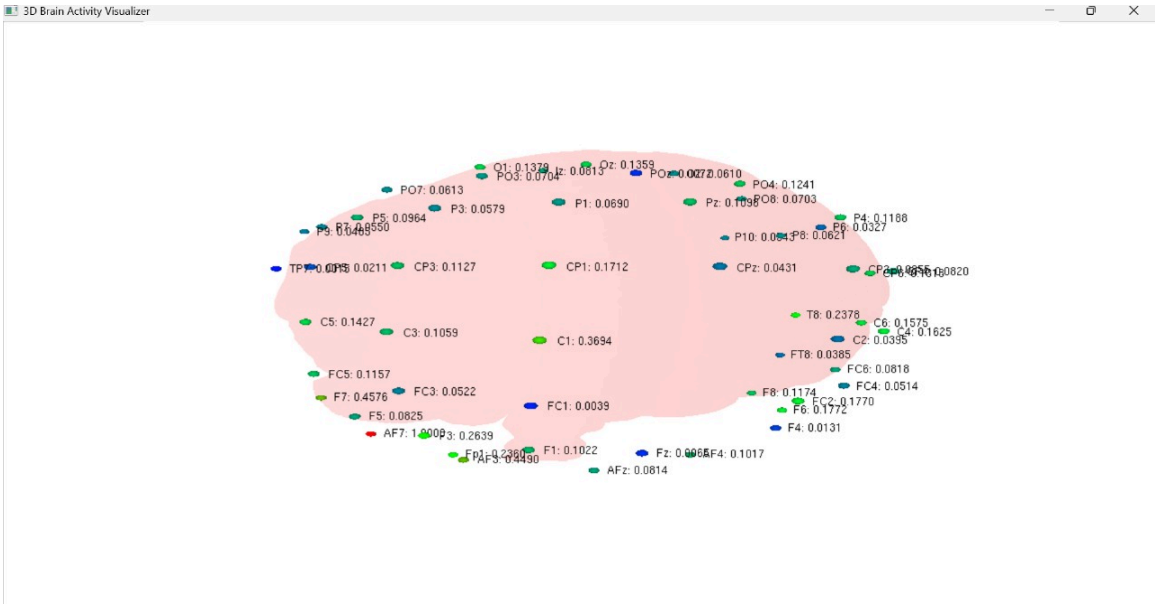


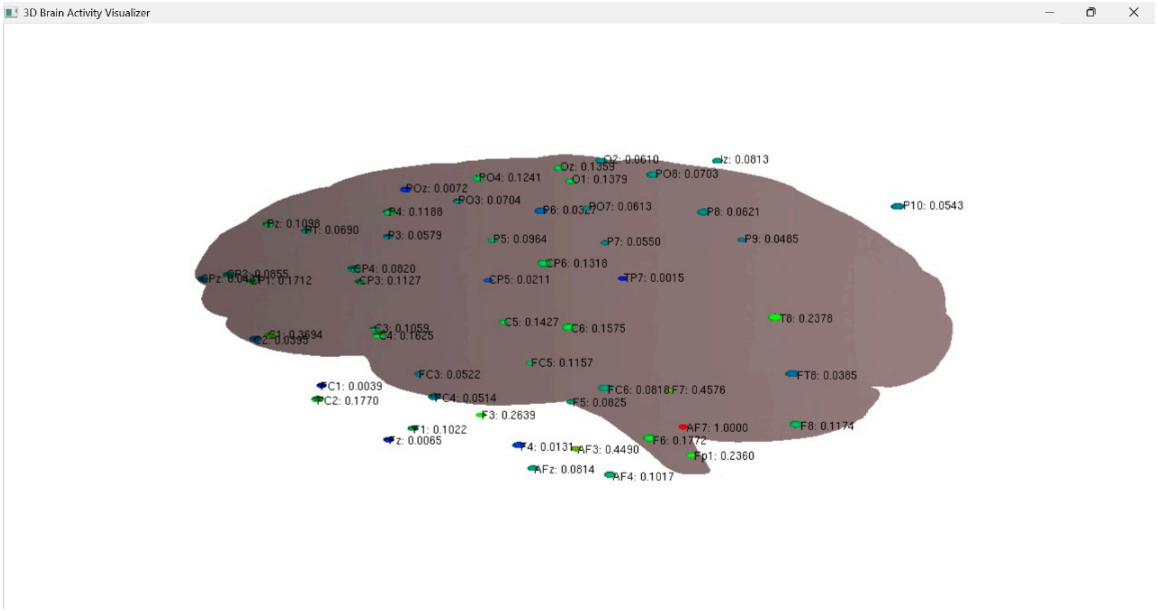
```
    GLfloat light_pos[] = { 0, 0, 2, 1 };
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos);

    GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };  // ambient component
    GLfloat diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };  // diffuse component
    GLfloat specularLight[] = { 0.2f, 0.2f, 0.2f, 1.0f }; // optional

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);

    glEnable(GL_COLOR_MATERIAL);
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
}

int main(int argc, char** argv) {
    loadBrainModel("models/Brain_Model.obj");
    loadElectrodePositions("electrode_positions.json");
    normalizeElectrodePositions();
    loadActivityValues("activity.json");

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1000, 800);
    glutCreateWindow("3D Brain Activity Visualizer");

    initOpenGL();
```

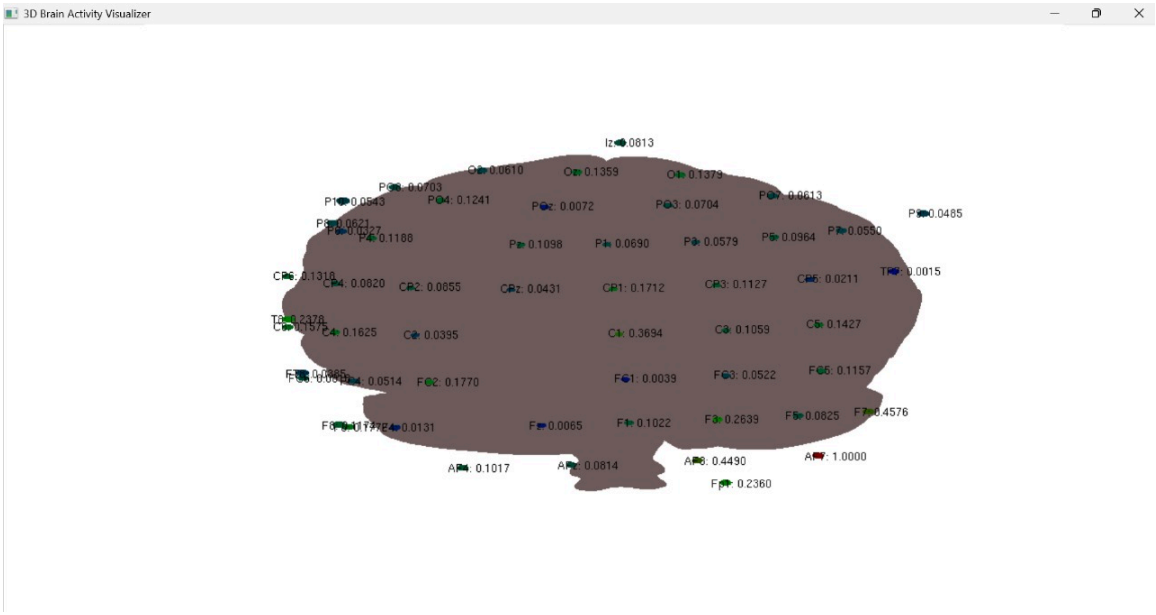


SCREENSHOTS




3D Brain Activity Visualizer
O1: 0.1379
Oz: 0.1359
PO3: 0.0704
Iz: 0.0813
PO4: 0.1241
PO7: 0.0613
P1: 0.0690
Pz: 0.1098
PO8: 0.0703
P3: 0.0579
P5: 0.0964
P4: 0.1188
P6: 0.0327
P8: 0.0621
P10: 0.0543
CP3: 0.1127
CP1: 0.1712
CPz: 0.0431
C5: 0.1427
C3: 0.1059
C1: 0.3694
T8: 0.2378
C6: 0.1575
C4: 0.1625
C2: 0.0395
FT8: 0.0385
FC6: 0.0818
FC5: 0.1157
FC3: 0.0522
F7: 0.4576
FC4: 0.0514
FC1: 0.0039
FC2: 0.1770
F8: 0.1174
F6: 0.1772
F5: 0.0825
F4: 0.0131
AF7: 1.0000
F3: 0.2639
F1: 0.1022
Fz: 0.0065
AF4: 0.1017
AF3: 0.4490
AFz: 0.0814


3D Brain Activity Visualizer
O2: 0.0610
Iz: 0.0813
Oz: 0.1359
PO4: 0.1241
O1: 0.1379
PO8: 0.0703
POz: 0.0072
PO3: 0.0704
P10: 0.0543
Pz: 0.1098
P4: 0.1188
P6: 0.0327
PO7: 0.0613
P8: 0.0621
P1: 0.0690
P3: 0.0579
P5: 0.0964
P7: 0.0550
P9: 0.0485
CP6: 0.1318
CP4: 0.0820
CP3: 0.1127
CP5: 0.0211
TP7: 0.0015
T8: 0.2378
C5: 0.1427
C6: 0.1575
C4: 0.1625
FC5: 0.1157
FC3: 0.0522
FT8: 0.0385
C1: 0.0039
FC6: 0.0818
F7: 0.4576
FC2: 0.1770
FC4: 0.0514
F5: 0.0825
F3: 0.2639
AF7: 1.0000
F8: 0.1174
F1: 0.1022
F6: 0.1772
Fz: 0.0065
F4: 0.0131
AF3: 0.4490
Fp1: 0.2360
AFz: 0.0814
AF4: 0.1017



REFERENCES

1. Dataset: https://orda.shef.ac.uk/articles/dataset/EEG_Data_for_Electrophysiological_signatures_of_brain_aging_in_autism_spectrum_disorder_/16840351

2. Main Research paper: https://www.sciencedirect.com/science/article/pii/S0010945222000090?via%3Dihub