Appendix C. Supplementary data

The following is the supplementary data to this article:

**Supplementary Fig. S1.** OpenGL rendering modes. (A) Immediate mode. Every time that the rendering command is called, all data has to be transferred from the CPU memory to the GPU memory. Therefore rendering performance scales $\sim O(n)$. (B) Vertex Buffer Object (VBO) mode. GPU memory can be allocated by the application and rendering occurs directly from GPU memory. Only data that changes has to be transferred from CPU to GPU. Therefore rendering performance is $\sim O(1)$.

**Supplementary Fig. S2.** Location of implanted electrodes, marked by an electrolytic lesion (arrow nr 1, white patch) and fluorescent dye staining (Dil, red) in one Animal (D38, see Table 2). Fluorescence is superimposed from the immediately neighboring slice. The visual areas D1/D2 are located in the dorso-medial and dorsal cortex, for which the ridge (arrow nr 2) served as an anatomical landmark during surgery. Electrodes were coated shortly with Dil dissolved in Ethanol before implantation. Slices were cut 60um thick and consecutive. Staining is for either Nissl or Fluorescence/DAPI for each other slide, so that the Nissl stain shown is from the slide immediately next to the Dil slide. Anatomical location abbreviations are (same notation as (Powers and Reiner, 1980), comparing to position A3.4 in the atlas): cm cortex medial, cdm cortex dorsomedial, cd cortex dorsal, DVR dorsal ventricular ridge, cp cortex periform.
### A)

**CPU memory**

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>X2</th>
<th>Y2</th>
<th>X3</th>
<th>Y3</th>
<th>X4</th>
<th>Y4</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>Xn</th>
<th>Yn</th>
</tr>
</thead>
</table>

![Diagram of CPU memory]

**GPU memory**

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>X2</th>
<th>Y2</th>
<th>X3</th>
<th>Y3</th>
<th>X4</th>
<th>Y4</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>Xn</th>
<th>Yn</th>
</tr>
</thead>
</table>

```python
# Code for drawing

glBegin(GL_LINE_STRIP)
for j in range(nbrPoints):
    glVertex2f(x[j], y[j])
glEnd()
```

~ $O(n)$

### B)

**CPU memory**

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>X2</th>
<th>Y2</th>
<th>X3</th>
<th>Y3</th>
<th>X4</th>
<th>Y4</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>Xn</th>
<th>Yn</th>
</tr>
</thead>
</table>

![Diagram of CPU memory]

**GPU memory**

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>X2</th>
<th>Y2</th>
<th>X3</th>
<th>Y3</th>
<th>X4</th>
<th>Y4</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>Xn</th>
<th>Yn</th>
</tr>
</thead>
</table>

```python
# Data preparation

glBindBuffer(GL_ARRAY_BUFFER, vbo_id)
glBufferSubData(GL_ARRAY_BUFFER, offset, sizeof(data), data)
glVertexPointer(nCoordinatesPerVertex, GL_FLOAT, stride, offset)
glDrawArrays(GL_LINE_STRIP, stride, nbrPoints)
```

~ $O(1)$