Commit 93a7256

Browse files

torvalds committed 5 days ago

```
Merge branch 'antigravity'

This is Google Antigravity fixing up my visualization tool (which was
also generated with help from google, but of the normal kind).

It mostly went smoothly, although I had to figure out what the problem
with using the builtin rectangle select was.  After telling antigravity
to just do a custom RectangleSelector, things went much better.

Is this much better than I could do by hand? Sure is.

* antigravity:
  Refine visualization: Zoom, Shortcuts, and Layout
  Implement RectangleSelector for custom zoom and fix whitespace
  visualize: fix navigation fighting with manual zoom
  visualize: implement independent keyboard navigation
  visualize: add keyboard navigation (arrows/space)
  visualize: show individual sample markers at high zoom levels
  visualize: refine zoom behavior
  visualize: improvement to memory usage and navigation
```

main 2 parents 4e52425 + 4e3eb84 commit 93a7256

Filter files...

visualize.py 1

1 file changed +331 -132 lines changed

Search within code

+331 -132

visualize.py

```
@@ -2,174 +2,373 @@
 2   2  import numpy as np
```

Comment on line R2

hoangdesu 14 hours ago

the beginning of something new

```
 3   3  import matplotlib.pyplot as plt
 4   4  import matplotlib.ticker as ticker
     5  + from matplotlib.widgets import Slider,
        RectangleSelector
     6  +
 5   7  import os
 6      - from functools import partial
 7   8  import argparse
 8   9
 9      - # I should probably use some actual audio library
        like librosa
10      - # but this is numpy + matplotlib and google helping
        me
11      -
12  10  # --- Constants ---
13      - CHUNK_DURATION_SEC = 2.0
14      - STEP_DURATION_SEC = 1.0
    11  + INITIAL_WINDOW_SEC = 2.0
15  12  BYTES_PER_SAMPLE = 4
16      - MAX_Y_LIMIT_HARD = 1.0  # Hard Limit for full scale
        unity
17      - MIN_Y_LIMIT_HARD = -1.0
18      - MIN_Y_ZOOM_DEFAULT = 0.5  # Default minimum Y view
        range (-0.5 to 0.5)
19      -
20      - # --- Global State ---
21      - file_handles = []
22      - current_offset_bytes = 0
23      - current_chunk_samples = 0
24      - SAMPLE_RATE_HZ = 48000
25      -
```

```
26        - def zoom_handler(event, ax):
27        -     """Handles X/Y scroll zoom centered on cursor,
          constrained to current 4s chunk."""
28        -     if event.inaxes is None: return
29        -
30        -     scale_factor = 0.8 if event.button == 'up' else
          1.2
31        -
32        -     # X-Axis limits (0 to CHUNK_DURATION_SEC)
33        -     xlim = ax.get_xlim()
34        -     cur_width = xlim[1] - xlim[0]
35        -     new_width = cur_width * scale_factor
36        -
37        -     # arbitrary X limit: 100 samples
38        -     new_width = max(100 / SAMPLE_RATE_HZ, new_width)
39        -
40        -     rel_pos_x = (event.xdata - xlim[0]) / cur_width
          if cur_width > 0 else 0.5
41        -     new_left = event.xdata - (new_width * rel_pos_x)
42        -     new_right = new_left + new_width
43        -
44        -     ax.set_xlim([max(0, new_left),
          min(CHUNK_DURATION_SEC, new_right)])
45        -
46        -     # Y-Axis limits (-1.0 to 1.0)
47        -     ylim = ax.get_ylim()
48        -     cur_height = ylim[1] - ylim[0]
49        -     new_height = cur_height * scale_factor
50        -
51        -     # arbitrary Y limit: 1/10th of the scale
52        -     new_height = max(0.2, new_height)
      13  + MAX_WIDTH_SEC = 2.0  # Max zoom out
      14  +
      15  + class WaveformVisualizer:
      16  +     def __init__(self, filenames, rate,
          min_zoom_samples=100):
      17  +         self.rate = rate
      18  +         self.navigating = False
      19  +         self.filenames = filenames
      20  +         self.min_zoom_samples = min_zoom_samples
      21  +         self.min_width_sec = min_zoom_samples / rate
      22  +
      23  +         self.mapped_files = []
      24  +         self.lines = []
      25  +         self.max_samples = 0
      26  +
      27  +         # Load files
      28  +         for f in filenames:
      29  +             try:
      30  +                 fsize = os.path.getsize(f)
      31  +                 samples = fsize // BYTES_PER_SAMPLE
      32  +                 mm = np.memmap(f, dtype=np.int32,
          mode='r', shape=(samples,))
      33  +                 self.mapped_files.append((mm,
          os.path.basename(f)))
      34  +                 self.max_samples =
          max(self.max_samples, samples)
      35  +             except Exception as e:
      36  +                 print(f"Error opening {f}: {e}")
      37  +
      38  +         if not self.mapped_files:
      39  +             return
      40  +
      41  +         self.duration_sec = self.max_samples /
          self.rate
      42  +         self.setup_ui()
      43  +
      44  +     def setup_ui(self):
      45  +         self.fig, self.ax = plt.subplots(figsize=(12,
          6))
      46  +         # Manual "tight layout" to maximize space but
          keep room for slider
      47  +         plt.subplots_adjust(left=0.05, right=0.95,
          top=0.95, bottom=0.15)
```

```
+
+        # Create line objects
+        for _, name in self.mapped_files:
+            line, = self.ax.plot([], [], linewidth=0.8, label=name)
+            self.lines.append(line)
+
+        self.ax.grid(True, which='both', linestyle=':', alpha=0.5)
+        self.ax.set_xlabel("Time (s)")
+        self.ax.set_ylabel("Amplitude")
+        self.ax.legend(loc='upper right', fontsize='x-small')
+
+        # Slider setup
+        ax_slider = plt.axes([0.15, 0.05, 0.7, 0.03])
+        self.slider = Slider(ax_slider, 'Time', 0, self.duration_sec, valinit=0)
+        self.slider.on_changed(self.update_slider)
+
+        # Scroll Zoom setup
+        self.fig.canvas.mpl_connect('scroll_event', self.on_scroll)
+        self.fig.canvas.mpl_connect('key_press_event', self.on_key)
+        self.ax.callbacks.connect('xlim_changed', self.on_xlim_changed)
+
+        # Custom Rectangle Selector
+        self.rs = RectangleSelector(
+            self.ax, self.on_select,
+            useblit=True,
+            button=[1],  # Left mouse button
+            minspanx=5, minspany=5,
+            spancoords='pixels',
+            interactive=False
+        )
+
+        # Initial View
+        self.update_view(0, INITIAL_WINDOW_SEC)
+
+        plt.show()
+
+    def get_chunk(self, start_time, window_duration):
+        start_sample = int(start_time * self.rate)
+        # Ensure we don't request negative start
+        start_sample = max(0, start_sample)
+
+        end_sample = int((start_time + window_duration) * self.rate)
+
+        global_min_y, global_max_y = 1e9, -1e9
+        has_data = False
+
+        for line, (mm, _) in zip(self.lines, self.mapped_files):
+            if start_sample >= mm.size:
+                line.set_data([], [])
+                continue
+
+            safe_end = min(end_sample, mm.size)
+            if safe_end <= start_sample:
+                 line.set_data([], [])
+                 continue
+
+            chunk = mm[start_sample:safe_end]
+
+            if chunk.size > 0:
+                normalized = chunk.astype(np.float32) / 2147483648.0
+
+                # Precise time axis using arange
+                t_origin = start_sample / self.rate
```

```
111 +                 t_axis = np.arange(chunk.size) / self.rate + t_origin
112 +
113 +                 line.set_data(t_axis, normalized)
114 +
115 +                 # Show markers if zoomed in enough (few samples visible)
116 +                 if chunk.size < 300:
117 +                     line.set_marker('.')
118 +                     line.set_markersize(3)
119 +                 else:
120 +                     line.set_marker("")
121 +
122 +                 global_min_y = min(global_min_y, np.min(normalized))
123 +                 global_max_y = max(global_max_y, np.max(normalized))
124 +                 has_data = True
125 +             else:
126 +                 line.set_data([], [])
127 +
128 +         return has_data, global_min_y, global_max_y
129 +
130 +     def update_view(self, start_time, width):
131 +         """Core update logic: loads data and sets limits (Constrained Mode)."""
132 +         if self.navigating: return
133 +         self.navigating = True
134 +         try:
135 +             # Clamp width
136 +             width = max(self.min_width_sec, min(width, MAX_WIDTH_SEC))
53  137
54      -     rel_pos_y = (event.ydata - ylim[0]) / cur_height if cur_height > 0 else 0.5
55      -     new_bottom = event.ydata - (new_height * rel_pos_y)
56      -     new_top = new_bottom + new_height
138 +             # Clamp start time
139 +             start_time = max(0, min(start_time, self.duration_sec))
57  140
58      -     # Constrain Y to hard limits
59      -     ax.set_ylim([max(MIN_Y_LIMIT_HARD, new_bottom), min(MAX_Y_LIMIT_HARD, new_top)])
60      -     ax.figure.tight_layout()
61      -     ax.figure.canvas.draw_idle()
141 +             has_data, min_y, max_y = self.get_chunk(start_time, width)
62  142
63      - def update_plot_chunk(direction, ax, fig):
64      -     """
65      -     Moves file pointer by 2s, reads a 4s chunk, and sets the Y-axis
66      -     based on the *actual* data range within that chunk.
67      -     """
68      -     global current_offset_bytes, current_chunk_samples
143 +             # IMPORTANT: Set limits on the MAIN axes, explicitly using self.ax
144 +             self.ax.set_xlim(start_time, start_time + width)
69  145
70      -     step_bytes = int(STEP_DURATION_SEC * SAMPLE_RATE_HZ * BYTES_PER_SAMPLE)
71      -     new_offset = current_offset_bytes + (direction * step_bytes)
146 +             # Tight Y-axis scaling logic
147 +             if has_data and max_y > min_y:
148 +                 # Symmetric zoom centered at 0
149 +                 max_val = max(abs(min_y), abs(max_y))
72  150
73      -     if new_offset < 0: new_offset = 0
```

```
-     current_offset_bytes = new_offset
+                 if max_val < 1e-6:
+                     max_val = 0.01
+                 else:
+                     max_val *= 1.05

-     ax.clear()
-     read_count = int(CHUNK_DURATION_SEC * SAMPLE_RATE_HZ)
+                 self.ax.set_ylim(-max_val, max_val)
+             else:
+                  # Default fallback if no data
+                  self.ax.set_ylim(-1.0, 1.0)

-     # Track the min/max of all loaded data for setting Y-axis limits
-     global_min_y, global_max_y = 1e9, -1e9
+             self.fig.canvas.draw_idle()
+         finally:
+             self.navigating = False

-     for handle, name in file_handles:
-         handle.seek(current_offset_bytes, os.SEEK_SET)
-         data = np.fromfile(handle, dtype=np.int32, count=read_count)
+     def update_slider(self, val):
+         """Callback for slider change."""
+         if self.navigating: return

-         if data.size > 0:
-             normalized = data / 2147483648.0
-             time_axis = np.arange(data.size) / SAMPLE_RATE_HZ
-             ax.plot(time_axis, normalized, linewidth=0.8, label=name)
-             current_chunk_samples = data.size
+         # Get current width from the main property
+         current_xlim = self.ax.get_xlim()
+         width = current_xlim[1] - current_xlim[0]

-             # Update the global min/max for this specific loaded chunk
-             global_min_y = min(global_min_y, normalized.min())
-             global_max_y = max(global_max_y, normalized.max())
+         # Fallback for init
+         if width <= 0: width = INITIAL_WINDOW_SEC

-     # Absolute time formatting
-     abs_offset_samples = current_offset_bytes // BYTES_PER_SAMPLE
-     def time_fmt(x, pos):
-         return f'{(x + abs_offset_samples / SAMPLE_RATE_HZ):.3f}s'
+         self.update_view(val, width)

-
ax.xaxis.set_major_formatter(ticker.FuncFormatter(time_fmt))
+     def on_scroll(self, event):
+         """Handle zoom."""
+         if event.inaxes != self.ax: return

-     # --- Y-Axis Auto-Ranging Logic ---
-     if global_max_y > global_min_y:
-         # Calculate a symmetric range around zero that encompasses the data
-         max_abs = max(abs(global_min_y), abs(global_max_y))
+         xlim = self.ax.get_xlim()
+         cur_width = xlim[1] - xlim[0]
```

```
108     -         # Default to a range of at least -0.5 to 0.5 (MIN_Y_ZOOM_DEFAULT)
109     -         # We ensure the limits are at least as wide as our minimum default zoom level
110     -         # and do not exceed the hard -1.0 to 1.0 limits.
111     -         view_limit = max(max_abs, MIN_Y_ZOOM_DEFAULT)
        185   +         scale_factor = 0.8 if event.button == 'up' else 1.2
        186   +         new_width = cur_width * scale_factor
112     187
113     -         # Add a little padding to the view_limit
114     -         view_limit *= 1.05
        188   +         # Clamp zoom
        189   +         new_width = max(self.min_width_sec, min(new_width, MAX_WIDTH_SEC))
115     190
116     -         # Clamp the view limits to the hard limits (-1.0 to 1.0)
117     -         view_limit = min(view_limit, MAX_Y_LIMIT_HARD)
        191   +         # Center zoom
        192   +         rel_x = (event.xdata - xlim[0]) / cur_width
        193   +         new_start = event.xdata - (new_width * rel_x)
118     194
119     -         ax.set_ylim(-view_limit, view_limit)
120     -     else:
121     -         # Fallback if chunk is empty or flatlined
122     -         ax.set_ylim(-MIN_Y_ZOOM_DEFAULT, MIN_Y_ZOOM_DEFAULT)
        195   +         # Clamp bounds
        196   +         new_start = max(0, min(new_start, self.duration_sec - new_width))
123     197
        198   +         # CRITICAL FIX: Set the plot limits FIRST
        199   +         self.ax.set_xlim(new_start, new_start + new_width)
124     200
125     -     # Reset X View (always 0 to 4s for this logic)
126     -     ax.set_xlim(0, CHUNK_DURATION_SEC)
127     -     ax.axhline(0, color='black', lw=0.5, ls='--')
128     -     ax.grid(True, which='both', linestyle=':', alpha=0.5)
129     -     ax.set_title(f"Position: {(abs_offset_samples/SAMPLE_RATE_HZ):.3f}s | Fs: {SAMPLE_RATE_HZ}Hz")
130     -     ax.set_xlabel("Time")
131     -     ax.set_ylabel("Amplitude")
132     -     ax.legend(loc='upper right', fontsize='x-small')
        201   +         # Then update slider
        202   +         self.slider.set_val(new_start)
133     203
134     -     fig.tight_layout()
135     -     fig.canvas.draw_idle()
        204   +     def on_xlim_changed(self, ax):
        205   +         """Handle external xlim changes (e.g. from Toolbar). Unconstrained load."""
        206   +         if self.navigating: return
        207   +         self.navigating = True
        208   +         try:
        209   +             xlim = ax.get_xlim()
        210   +             start_time = xlim[0]
        211   +             width = xlim[1] - xlim[0]
        212   +
        213   +             # Reload data (No constraints)
        214   +             self.get_chunk(start_time, width)
        215   +
        216   +             # Sync slider silently
        217   +             if hasattr(self, 'slider'):
        218   +                 old_eventson = self.slider.eventson
        219   +                 self.slider.eventson = False
        220   +                 self.slider.set_val(start_time)
        221   +                 self.slider.eventson = old_eventson
        222   +         finally:
```

```
223 +             self.navigating = False
224 +
225 +     def on_key(self, event):
226 +         """Handle keyboard shortcuts (Independent
    Navigation)."""
227 +         if self.navigating: return
228 +         self.navigating = True
229 +         try:
230 +             # Get properties
231 +             xlim = self.ax.get_xlim()
232 +             ylim = self.ax.get_ylim()
233 +
234 +             width = xlim[1] - xlim[0]
235 +             height = ylim[1] - ylim[0]
236 +
237 +             changed = False
238 +
239 +             if event.key == 'right':
240 +                 shift = width * 0.25
241 +                 self.ax.set_xlim(xlim[0] + shift,
    xlim[1] + shift)
242 +                 changed = True
243 +             elif event.key == 'left':
244 +                 shift = width * 0.25
245 +                 self.ax.set_xlim(xlim[0] - shift,
    xlim[1] - shift)
246 +                 changed = True
247 +             elif event.key == 'up':
248 +                 shift = height * 0.25
249 +                 self.ax.set_ylim(ylim[0] + shift,
    ylim[1] + shift)
250 +                 changed = True
251 +             elif event.key == 'down':
252 +                 shift = height * 0.25
253 +                 self.ax.set_ylim(ylim[0] - shift,
    ylim[1] - shift)
254 +                 changed = True
255 +             elif event.key == 'pagedown': # Zoom In
    (0.5x width)
256 +                 center = (xlim[0] + xlim[1]) / 2
257 +                 new_width = width * 0.5
258 +                 new_width = max(new_width,
    self.min_width_sec)
259 +                 new_start = center - (new_width / 2)
260 +                 # Clamp start
261 +                 new_start = max(0, min(new_start,
    self.duration_sec - new_width))
262 +                 self.ax.set_xlim(new_start, new_start
    + new_width)
263 +                 changed = True
264 +             elif event.key == 'pageup': # Zoom Out (2x
    width)
265 +                 center = (xlim[0] + xlim[1]) / 2
266 +                 new_width = width * 2.0
267 +                 new_width = min(new_width,
    MAX_WIDTH_SEC)
268 +                 new_start = center - (new_width / 2)
269 +                 # Clamp start
270 +                 new_start = max(0, min(new_start,
    self.duration_sec - new_width))
271 +                 self.ax.set_xlim(new_start, new_start
    + new_width)
272 +                 changed = True
273 +
274 +             if changed:
275 +                 # Reload data for new X view
276 +                 # We need new xlim
277 +                 new_xlim = self.ax.get_xlim()
278 +                 self.get_chunk(new_xlim[0],
    new_xlim[1] - new_xlim[0])
279 +
280 +                 # Sync slider silently
281 +                 if hasattr(self, 'slider'):
```

```
282 +                     old_eventson =
    self.slider.eventson
283 +                     self.slider.eventson = False
284 +                     self.slider.set_val(new_xlim[0])
285 +                     self.slider.eventson =
    old_eventson
286 +
287 +                 self.fig.canvas.draw_idle()
288 +
289 +         finally:
290 +              self.navigating = False
291 +
292 +         # Space Bar (Reset Zoom) - Only reset Y axis
    to fit visible data (Auto-scale)
293 +         if event.key == ' ':
294 +             xlim = self.ax.get_xlim()
295 +             width = xlim[1] - xlim[0]
296 +             self.update_view(xlim[0], width)
297 +
298 +     def on_select(self, eclick, erelease):
299 +         """Handle rectangle selection."""
300 +         if self.navigating: return
301 +
302 +         # Calculate new ranges from the selection
303 +         x1, y1 = eclick.xdata, eclick.ydata
304 +         x2, y2 = erelease.xdata, erelease.ydata
305 +
306 +         start_time = min(x1, x2)
307 +         end_time = max(x1, x2)
308 +         width = end_time - start_time
309 +
310 +         # Enforce minimum width
311 +         if width < self.min_width_sec:
312 +             center = (start_time + end_time) / 2
313 +             width = self.min_width_sec
314 +             start_time = center - width / 2
315 +
316 +         # Clamp start time
317 +         start_time = max(0, min(start_time,
    self.duration_sec - width))
318 +
319 +         # Update view (which reloads data)
320 +         # Note: update_view handles height scaling
    automatically based on data,
321 +         # but if we want to respect the user's Y
    selection we might need to override.
322 +         # However, for audio, usually we want to see
    the full amplitude or auto-scaled.
323 +         # The user asked for "arbitrary zooming be
    implemented", which implies Y might be important.
324 +         # But our `update_view` forces Y symmetry
    currently.
325 +         # Let's trust `update_view` for now as it
    handles the complexity of data loading.
326 +         # If we really want to zoom to the Y
    rectangle, we would set ylim manually.
327 +         # Given the previous context was about "Zoom
    to Rectangle" interfering with "movement and zoom
    control",
328 +         # let's try to match the "Zoom to Rectangle"
    behavior which DOES set specific X/Y limits.
329 +
330 +         # However, our update_view logic re-calculates
    Y based on data in the new X range...
331 +         # Let's modify the flow slightly: use
    update_view for X, then apply Y if it makes sense?
332 +         # Actually, standard audio visualization often
    keeps Y symmetric or 0-1.
333 +         # If the user draws a small box on a peak,
    they expect to zoom into that peak (Y-wise too).
334 +
335 +         # So let's set limits directly, similar to
    on_scroll/on_key, but for both axes.
336 +
```

```
                +         if self.navigating: return
                +         self.navigating = True
                +         try:
                +             # X Axis Logic
                +             new_width = max(self.min_width_sec,
                  min(width, MAX_WIDTH_SEC))
                +             # If user selected < min width, we
                  centered it above.
136             343
137             -   def setup_app(filenames, rate):
138             -       global file_handles, SAMPLE_RATE_HZ
139             -       SAMPLE_RATE_HZ = rate
                344 +             self.get_chunk(start_time, new_width)
140             345
141             -       for f in filenames:
142             -           try:
143             -               file_handles.append((open(f, 'rb'),
                  os.path.basename(f)))
144             -           except Exception as e:
145             -               print(f"Error opening {f}: {e}")
                346 +             self.ax.set_xlim(start_time, start_time +
                  new_width)
146             347
147             -       if not file_handles: return
                348 +             # Y Axis Logic
                349 +             y_min = min(y1, y2)
                350 +             y_max = max(y1, y2)
148             351
149             -       fig, ax = plt.subplots(figsize=(12, 6))
                352 +             self.ax.set_ylim(y_min, y_max)
150             353
151             -       # Event Connections
152             -       fig.canvas.mpl_connect('scroll_event',
                  partial(zoom_handler, ax=ax))
153             -       fig.canvas.mpl_connect('key_press_event', lambda
                  e:
154             -           update_plot_chunk(1, ax, fig) if e.key ==
                  'right' else
155             -           update_plot_chunk(-1, ax, fig) if e.key ==
                  'left' else None)
                354 +             self.fig.canvas.draw_idle()
156             355
157             -       # Initial Render
158             -       update_plot_chunk(0, ax, fig)
                356 +             # Sync Slider
                357 +             if hasattr(self, 'slider'):
                358 +                 old_eventson = self.slider.eventson
                359 +                 self.slider.eventson = False
                360 +                 self.slider.set_val(start_time)
                361 +                 self.slider.eventson = old_eventson
159             362
160             -       # Enable "Zoom to Rectangle" by default
161             -       try:
162             -           fig.canvas.toolbar.zoom()
163             -       except:
164             -           pass
                363 +         finally:
                364 +             self.navigating = False
165             365
166             -       plt.show()
167             -       for h, _ in file_handles: h.close()
168             366
169             367   if __name__ == "__main__":
170             -       parser =
                  argparse.ArgumentParser(description="Linux Audio
                  Waveform Visualizer 2026")
                368 +     parser =
                  argparse.ArgumentParser(description="Linux Audio
                  Waveform Visualizer 2026 (mmap)")
171             369       parser.add_argument('files', nargs='+',
                  help="Input .bin files (int32)")
172             370       parser.add_argument('--rate', type=int,
                  default=48000, help="Sample rate (Hz)")
```

```
      371   +     parser.add_argument('--min-zoom-samples',
                type=int, default=100, help="Minimum samples to show
                when zoomed in")
173   372         args = parser.parse_args()
174         -     setup_app(args.files, args.rate)
175   373
      374   +     app = WaveformVisualizer(args.files, args.rate,
                args.min_zoom_samples)
```

Comments 118

Load more comments

hiratazx yesterday

whoa

Mr-Rahul-Paul yesterday

let this be known, I was here

gempir yesterday

Some Google exec is screaming full of joy and excitement right now that Linus somehow stumbled into using Antigravity over all the better competitors, which there are many of.

0xanis yesterday

andrzejoblong yesterday

Thank you for your hard work over the past years. It has been a pleasure to witness the impact you have had on the open source world.

jknppp yesterday

👌😎🔫

floe yesterday

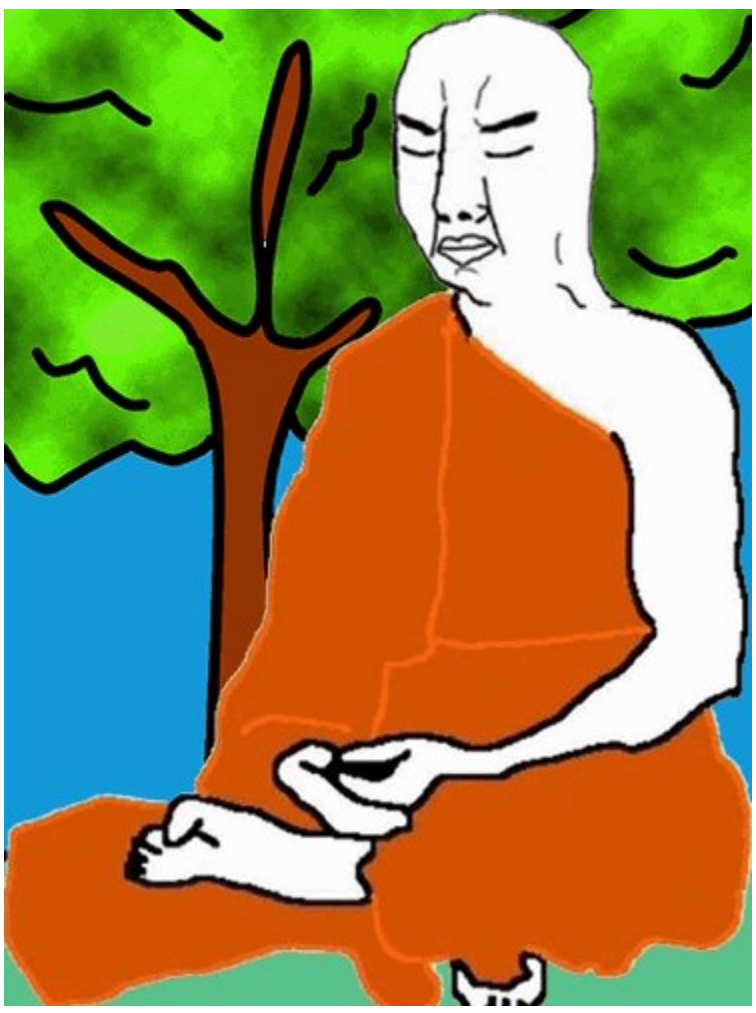
Quod licet Iovi, non licet bovi 😉

furkangr yesterday

Witnessing AI writes a chunk of code in the twinkling of an eye was less shocking than Torvalds complimenting it, which feels like witnessing the assassination of Archduke Franz Ferdinand. Sure, we better evolve with AI or collapse in the darkness. Maybe the role of swe is evolving, less code writing and more architecting, product management comes next. It's not only non-technicals getting hands on tech, also the opposite 😉

Abdspmask yesterday

Don't cry because it's over



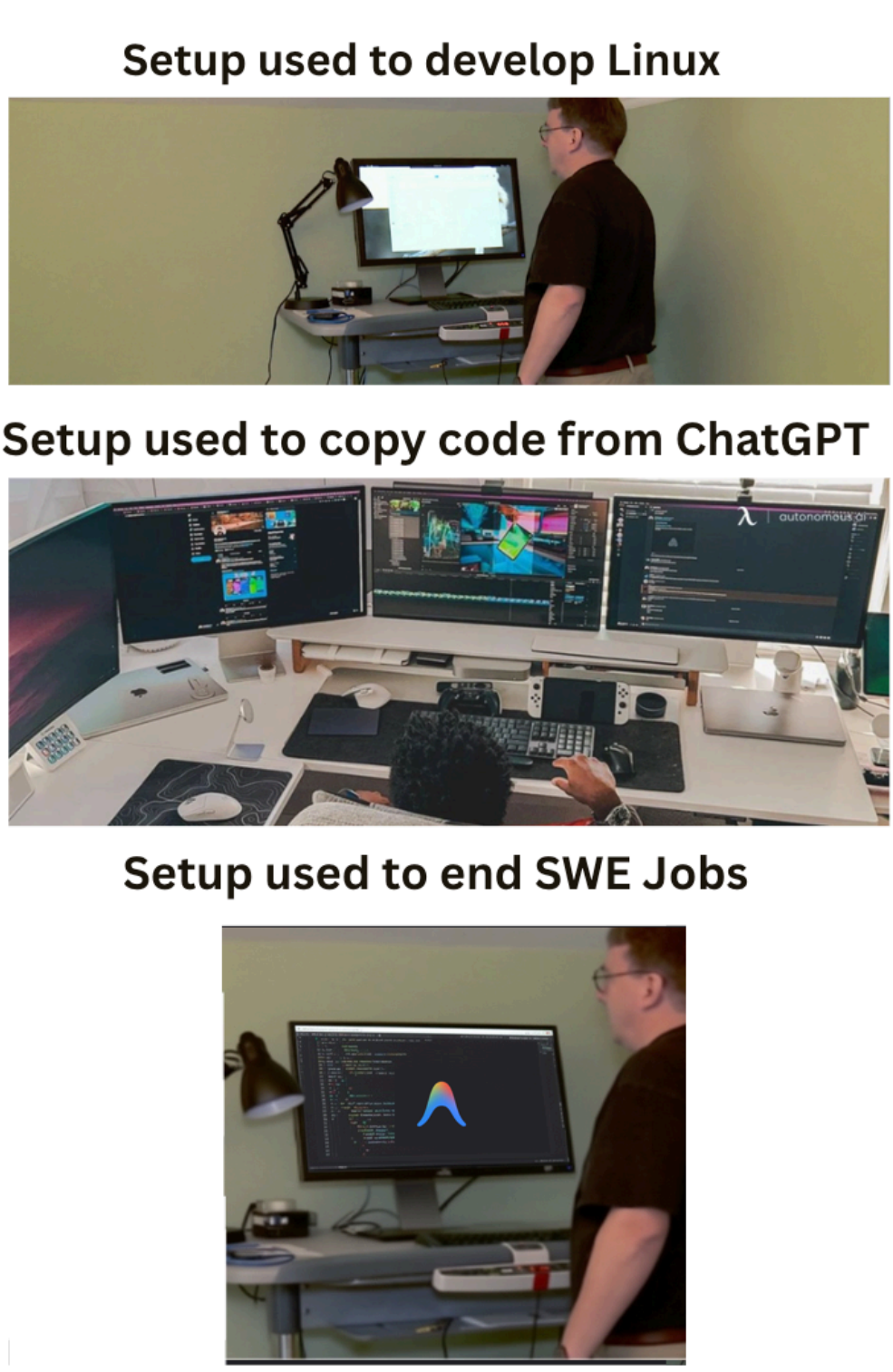
be happy because it even happened

RidaHajjAli yesterday

Rqcker yesterday

historic moment

gianpaj yesterday



henriacle 20 hours ago

i'm so glad that my plumber skills are ok.

sjkim1127 20 hours ago

He'll be fine—he's a real engineer, not just some cookie-cutter app dev.

tillua467 18 hours ago

I feel like ai should be used to Improve the code Little or Help with stuff That Normally we have low knowledge, I mean he is not vibe coding the entire project so it's justified in my opinion

abdullah-al-mridul 18 hours ago

rewyekha 18 hours ago

Over ? -_-!

crajycommits 18 hours ago

I should frame this COMMIT and put it on my wall.

janos 17 hours ago

I will sleep much better now.

0x00nier 17 hours ago

I can understand using it on Python scripts tbf

nuriofernandez 16 hours ago

2026 begins

bienvenudev 16 hours ago

Marking this spot in history 📌

Ndifreke 16 hours ago

😱

cxinu 16 hours ago

ivanm07 16 hours ago

I was here

felipenmoura 16 hours ago

Using AI to speed stuff up, specially when you already know about what and how you want it to be built, is really useful.

Please notice he did not say he *vibe coded* it ... he knew what was going on, there :)

Charan-gdev 15 hours ago

Lmaoo a nice start to the year

AperanoGavin 15 hours ago

i guess i have to use it too

umeshgmrl 15 hours ago

Future me, this was it.

cya 14 hours ago

> Is this much better than I could do by hand? Sure is.

RodricBr 14 hours ago

If the creator himself uses it, who are we - mere mortals - to deny such an act?

hoangdesu 14 hours ago

Feli87 14 hours ago

welcome to the hell xD

emirkabal 13 hours ago

👌

lilazero 13 hours ago

Damn. Hi history books. Hopefully I survive the cyborg uprising

proailurus 13 hours ago

put me in the screenshot

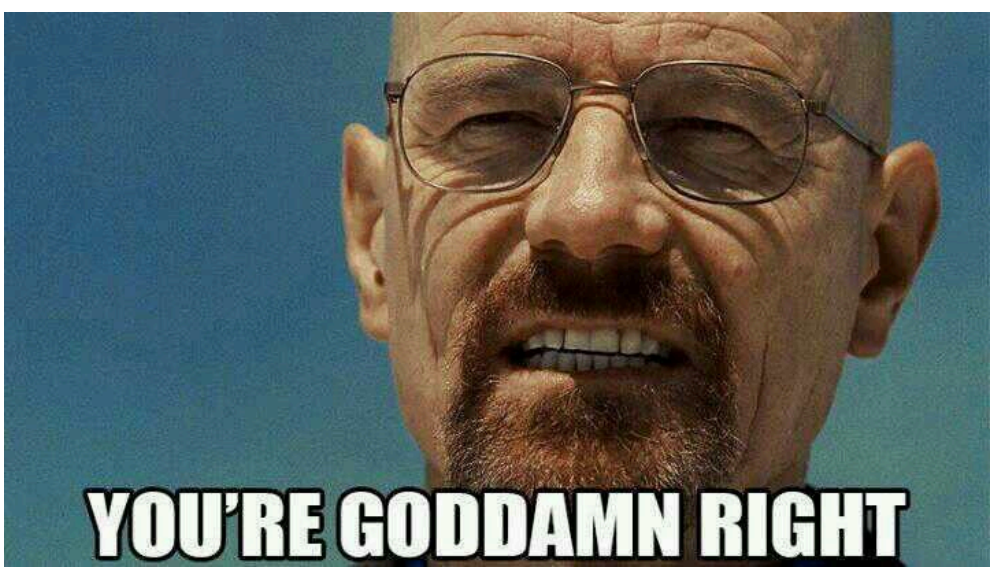
tonymishler 13 hours ago

It was a good run, lads

Lhesnor 12 hours ago

nah, from now on, I give up

ofurkanuygur 12 hours ago



dn768 11 hours ago

My friend sent me a free trial of Claude a few weeks back. Maybe this paired with the StackOverflow graph tells me I have to at least give it a try... 😔

Lorenzo-Machado 10 hours ago

I don't know if I find SWE tuff anymore... I will become a carpenter, I guess

rollerozxa 10 hours ago

holy cuddles

Lorenzo-Machado 10 hours ago

> nah, from now on, I give up

same

Regularname11 10 hours ago

it's not "vibe coding" if you know what you are doing and what results you are expecting.. smh

fubarfubaz 7 hours ago

i was here

crosseax 7 hours ago

historical commit

fzlfade 6 hours ago

here i am, in historical event

Vjay15 6 hours ago

You guys still forget that this is a toy project of linus lol, and he is just trying to have a good time

matsyui 5 hours ago

Oh my god

jayshrivastava0 5 hours ago

And Yes kidss.....I have lived through

1. 2012 - When world was ending
2. 2020 - World Ending Pandemic
3. 2026 - The FIRST real Software Engineer used AI for coding.

akellysoft 5 hours ago

I want to join this legendary moment also. :)

KaunAnkit 3 hours ago

linus

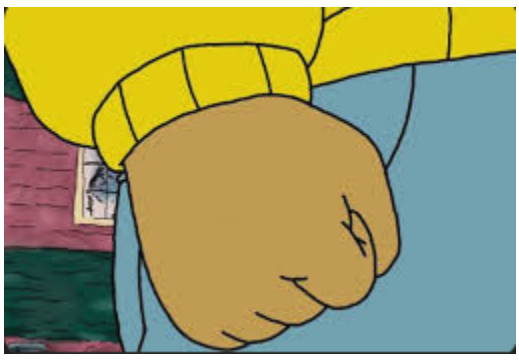
onikuwo835 3 hours ago

If even the world's best programmer doesn't want to write Python, why do we still call Python a programming language?

keymastervn 3 hours ago

This is a legendary moment

sewandev 3 hours ago



Jerry-Terrasse 3 hours ago

It is the best of times, it is the worst of times.

JoeBrar 2 hours ago

Incredible. I was here

felipe-tactile 2 hours ago

I also want to be part of this, a new era

nxdun 1 hour ago

Palkiaa 23 minutes ago

cant wait for all the news articles announcing linus embracing AI

Please sign in to comment.