

Atelier Processing : typographie

fontViewer.pde

```
/*  
 *****  
  Nécessite l'installation de la librairie "Drop"  
 *****  
 */  
  
import java.awt.Font;  
import drop.*;  
  
SDrop drop;  
  
PFont defaultFont;  
PFont font;  
Font nativeFont;  
int size = 32;  
int margin = 2;  
int old_width, old_height;  
  
ArrayList<MyGlyph> glyphs = new ArrayList();  
int iFirst, iLast;  
float yOffset = 0;  
float yVel = 0;  
float yMax = 0;  
int selected = -1;  
//ArrayList  
  
void setup() {  
  size(800, 600);  
  //surface.setResizable(true);  
  //fullScreen();  
  old_width = width;  
  old_height = height;  
  
  printArray(PFont.list());  
  
  // Drag & drop  
  drop = new SDrop(this);  
  
  text(' ', 0, 0); // dirty hack to get processing's default font  
  defaultFont = g.textFont;  
  
  //loadFontFile("NotoEmoji-VariableFont_wght.ttf");  
}  
  
void loadFontFile(String fontFile) {  
  font = createFont(fontFile, size);  
  nativeFont = (Font) font.getNative();  
  textFont(font);  
  glyphs.clear();  
  selected = -1;  
  
  //int x = margin;  
  //int y = font.getSize() + margin;  
  int numChars = 0;  
  for (int i = 0; i < 0x10ffff; i++) {  
    if (nativeFont.canDisplay(i)) {  
      numChars++;  
      MyGlyph glyph = new MyGlyph();  
      glyph.codepoint = i;  
      glyph.s = new String(Character.toChars(i));  
      glyph.w = textWidth(glyph.s);  
      glyph.h = font.getSize();  
      /*if (x + glyph.w + margin > width) {  
        x = margin;  
        y += font.getSize() + margin;  
      }  
      glyph.x = x;  
      glyph.y = y;  
      x += glyph.w + margin;*/  
      glyphs.add(glyph);  
    }  
  }  
  updateFont();  
  
  if (!glyphs.isEmpty())  
    yMax = max(0, glyphs.get(glyphs.size()-1).y + 4 * margin - height);  
}
```

```

println(numChars, "glyphs in font");
}

void updateFont() {
    int x = margin;
    int y = font.getSize() + margin;
    for (MyGlyph glyph : glyphs) {
        if (x + glyph.w + margin > width) {
            x = margin;
            y += font.getSize() + margin;
        }
        glyph.x = x;
        glyph.y = y;
        x += glyph.w + margin;
    }
}

void draw() {
    background(255);

    if (font == null) {
        fill(0);
        textSize(32);
        String mess = "Glissez-déposez une font dans la fenêtre";
        float w = textWidth(mess);
        text(mess, (width - w) * 0.5f, height/2);
        return;
    }

    yOffset = constrain(yOffset + yVel, 0, yMax);
    yVel *= 0.9;

    pushMatrix();
    translate(0, -yOffset);

    textFont(font);
    fill(0);
    iFirst = glyphs.size();
    iLast = 0;
    for (int i = 0; i < glyphs.size(); i++) {
        MyGlyph glyph = glyphs.get(i);
        if (glyph.y > yOffset && glyph.y - glyph.h < height + yOffset) {
            if (i == selected)
                fill(255, 0, 0);
            else
                fill(0);
            text(glyph.s, glyph.x, glyph.y);
            if (i < iFirst)
                iFirst = i;
            if (i > iLast)
                iLast = i;
        }
    }

    if (selected >= 0) {
        MyGlyph glyph = glyphs.get(selected);
        textFont(defaultFont);
        textSize(18);
        String text = String.format("%d | 0x%h", glyph.codepoint, glyph.codepoint);
        float tw = textWidth(text);
        float th = defaultFont.getSize();
        fill(255, 220);
        noStroke();
        rect(glyph.x + 0.5*(glyph.w - tw), glyph.y + 4, tw + 16, th + 8, 8);
        fill(0, 0, 255);
        text(text, glyph.x + 0.5*(glyph.w - tw) + 8, glyph.y + 4 + th + 4);
    }

    popMatrix();
}

void mousePressed() {
    if (font == null)
        return;

    selected = -1;

    if (keyPressed && keyCode == 16)
        println("maj");

    for (int i = iFirst; i <= iLast; i++) {
        MyGlyph glyph = glyphs.get(i);
        if (mouseX > glyph.x && mouseX < glyph.x + glyph.w &&
            mouseY + yOffset > glyph.y - glyph.h && mouseY + yOffset < glyph.y) {
            selected = i;
            println(glyph.s);
            break;
        }
    }
}

```

```
void mouseReleased() {
  if (width != old_width || height != old_height) {
    updateFont();
    println("update");
    old_width = width;
    old_height = height;
  }
}

void mouseWheel(MouseEvent event) {
  yVel += 2 * event.getCount();
}

class MyGlyph {
  public String s;
  public int codepoint;
  public int x, y;
  public float w, h;
}

void dropEvent(DropEvent theDropEvent) {
  if (theDropEvent.isFile() && (theDropEvent.toString().endsWith(".ttf") || theDropEvent.toString().endsWith(".otf"))) {
    loadFontFile(theDropEvent.toString());
  }
}
```

Article extrait de : <https://lesporteslogiques.net/wiki/> - **WIKI Les Portes Logiques**

Adresse : https://lesporteslogiques.net/wiki/ressource/code/processing/atelier_typo?rev=1662652866

Article mis à jour: **2022/09/08 18:01**