Lo scriba A del *Codex Sinaiticus* e il font *Simonides*

Claudio Vincoletto, Massimiliano Dominici

Sommario

L'automazione del tratto calligrafico è uno dei sogni ricorrenti della tipografia digitale. Il percorso qui intrapreso ne rappresenta una delle più recenti incarnazioni, sebbene si configuri anche come uno specchio del passato: il respiro dello scriba ritorna a vivere, palpitando sulla pagina stampata.

Abstract

The calligraphic stroke automation is one of the recurring dreams of digital typography. The path taken here represents one of the most recent incarnations, though it also functions as a mirror of the past: the scribe's breath returns to life, vibrating on the printed page.

1 *Envoi*

Tutto ciò che rimanda al termine *traduzione*, nel suo significato etimologico di "portare oltre, tramandare", può rappresentare lo sfondo necessario sul quale si dipana l'insieme dei propositi che hanno guidato l'indagine di cui si vuole esporre brevemente.

L'essere postumo della scrittura è connaturato alla trasmissione della memoria umana alle nuove generazioni, l'estremo tentativo di fissare la precarietà del presente in una sopravvivenza artificiale che sfida i limiti insuperabili e irreversibili della morte. Una sottile traccia della vita dello scriba permane fisicamente nei segni grafici, e soprattutto nella loro riproducibilità, in grado di porre rimedio anche alla dissoluzione dei supporti materiali più durevoli.

La configurazione semiologica del testo si staglia irrimediabilmente sulla pagina, virtuale o immanente che sia, in cui viene a instaurare un insieme di relazioni cronotopiche. In tale campo di tensione si possono riconoscere alcuni elementi di *persistenza* e altri di *mutamento*.

Ogni tratto, ogni glifo, ogni riga ben composta, manifestano ordine e ripetizione. Senza un codice di riferimento condiviso non è possibile alcuna forma di comunicazione, e per essere assimilato il codice deve perdurare. Diversamente, l'impiego di tali elementi in una situazione più vasta e contingente richiede una sorta di adattabilità, di riconfigurazione, in cui si possono notare lievi variazioni di accomodamento delle regole. Questo è ciò che rimane della scrittura, un fluire costante di oscillazioni tra la competenza del saper scrivere e le trasformazioni in cui si risolve la sua stessa esecuzione.

La nostra ricerca si muove in tale ambito.

2 Il manoscritto

Un anno prima della prematura morte, nel lontano 1761, il naturalista Vitaliano Donati annotava nei suoi diari di aver ammirato, in seguito a una visita presso il monastero di Santa Caterina sul monte Sinai, «una Bibbia in membrane bellissime, assai grandi, sottili, e quadre, scritta in carattere rotondo e bellissimo». Già si rendeva conto di avere fra le mani un codice molto antico, almeno anteriore al settimo secolo.

Fu solo però nel 1844 che il filologo Constantine Tischendorf, assolutamente convinto dell'inestimabile valore del testo, riuscì a convincere i monaci del monastero affinché gli donassero alcune pagine del codice, che furono decifrate e successivamente pubblicate a Lipsia Tischendorf (1862). Qualche anno dopo, grazie all'intercessione dello zar Alessandro II di Russia, il volume fu trasportato alla Biblioteca Imperiale di San Pietroburgo. Solo nel 1933 trovò la sistemazione attuale presso la British Library.

Grazie a un intenso lavoro di digitalizzazione, il manoscritto è oggi consultabile online, arricchito di numerosi apparati la cui analisi accurata è stata fondamentale per la realizzazione del presente lavoro Codex Sinaiticus Project (2015).

Il *Codex Sinaiticus* risale probabilmente al quarto secolo e rappresenta uno dei più antichi codici in pergamena, si presume stilato ad Alessandria o Cesarea. Il testo presenta uno dei repertori più autorevoli e completi del Vecchio e del Nuovo Testamento, ma può anche considerarsi tra gli onciali greci biblici quello di più notevole pregio. È possibile distinguere la mano di tre scribi, malgrado la scrittura a calamo risulti essere piuttosto omogenea Knight (2009); Parker (2010).

Il *Simonides* prende spunto dallo scriba A, riproducendo nel dettaglio un carattere onciale biblico, unanimemente riconosciuto come elegante e prestigioso.

3 La lezione del *Punk Nova*

Con l'introduzione della tipografia digitale si pensò da subito che il computer fosse in grado di automatizzare una serie di processi in grado di simulare la complessità della calligrafia. Tuttavia le strategie messe in opera furono abbastanza limitate e ancora oggi molti strumenti vengono sottovalutati se non ignorati dalle grandi industrie del settore.

Tra i primi tentativi di realizzare un font i cui parametri sono progettati per dare vita a una compilazione variegata e dalle infinite possibilità di tra-

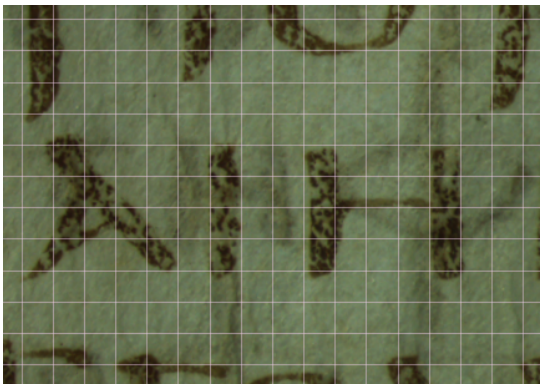


Figura 1: Particolare del *Codex Sinaiticus*: una griglia rivela il *modulo*.

sformazione figura in primo luogo il *Punk Font* di Donald Knuth Knuth (1988). Il disegno è originale (1985) e si ispira alle lezioni di Gerard and Marjan Unger. La natura stessa di METAFONT si rivela congeniale al progetto, proprio per la possibilità di organizzare i dati in modo flessibile.

Nel 2008 Taco Hoekwater e Hans Hagen, con l'idea di impiegare la tecnologia dei font `.otf` nel sistema ConTEXt, si cimentano con il tentativo di portare il *Punk Font* a tale formato, operazione tutt'altro che banale. Nasce l'esperienza del *Punk Nova* Hoekwater e Hagen (2008), un font corredato di una serie di glifi richiamati casualmente durante la compilazione del documento. In questo caso METAPOST viene usato in sostituzione di METAFONT, per la possibilità di generare i contorni di ogni glifo secondo le specifiche PostScript.

Uno script Python invoca FontForge che riassembla tutte le varianti generate per ogni glifo e predispone le funzioni che rendono casuale il principio di scelta nel corso della permutazione.

4 Dalla copia al modello

Al fine di ottenere delle varianti compatibili con l'originale è necessario immaginare un punto di partenza, una sorta di disegno ideale da cui derivano tutti gli altri. L'esame calligrafico del *Sinaiticus* non è semplice, i materiali fotografici utili allo scopo e forniti in rete non sono molti, e in questa fase non si è pensato di accedere direttamente al manoscritto. Con le immagini disponibili è comunque possibile individuare quel che si può definire un *modulo* di riferimento (figura 1).

Il tratteggio, per quanto irregolare, presenta dei punti focali facilmente individuabili nella sovrapposizione fotografica di un reticolato. Tale stratagemma è impiegato più frequentemente nell'analisi dei caratteri tipografici piuttosto che in paleografia. In ogni caso l'espediente si è rivelato efficace e sufficiente all'isolamento di un modulo, debitamente riportato su carta millimetrata.

Questa base, convertita in un sorgente per METAFONT, ha rappresentato lo stadio primordiale del progetto. Successivamente si è calcolato l'ambito in cui poteva variare ogni punto del tratto, circoscrivendo un campo di aleatorietà a sua volta ricondotto a coordinate spaziali.

La variabilità dei glifi è uno degli aspetti salienti della scrittura ed è stata implementata sfruttando alcune caratteristiche del programma di compilazione. L'introduzione di un'indeterminazione controllata è possibile con METAFONT in almeno due modalità: con `uniformdeviate` t viene fornito un numero u che è distribuito casualmente tra 0 e t; con `normaldeviate` invece ne risulta un numero casuale x tra zero e uno.

La strategia impiegata per il *Simonides* è la stessa usata da Knuth per il font *Punk* e qui se ne farà breve cenno. Si immagini uno spazio piano in cui ogni punto sia determinato dalle sue coordinate (x, y). Con il parametro `dev`, derivante dalla dimensione del glifo, si delimita un ambito di variazione che andrà a incidere sulla coordinata stessa.

```
hair_space# = 1/9 size#;
dev#:=1/12hair_space#;
```

Ridefinendo il comando `beginchar`, che stabilisce la larghezza, l'altezza e la profondità della scatola contenente il glifo, vengono inseriti due ulteriori valori numerici, uno per la componente orizzontale (h) l'altro per la verticale (v), i quali vengono ricalcolati in funzione della deviazione predisposta da `dev`.

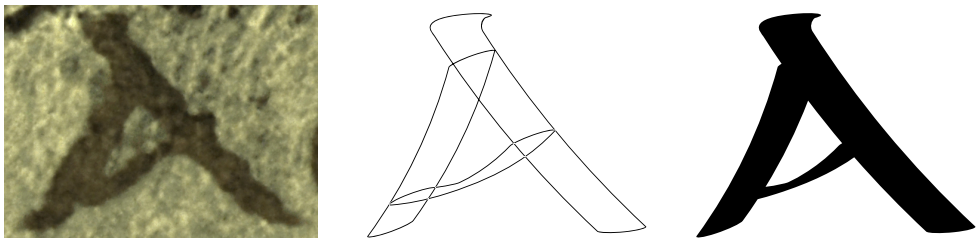
```
hdev := h * dev ;
vdev := v * dev ;
```

Infine una *macro* modifica il valore delle coordinate di un punto `z`, inserendo l'indice di casualità desiderata.

```
vardef pp expr z =
 z + (hdev * normaldeviate, vdev * normaldeviate)
 enddef;
```

Per quel che riguarda il disegno dell'alfabeto non ci si discosta molto da quello del greco classico, sebbene dotato di pochissimi accenti e allestito in almeno due corpi: uno più minuto per le abbreviazioni. È stata aggiunta una batteria di varianti ulteriori, per ampliare il criterio di scelta del compositore. Quest'ultima si è rivelata necessaria, in quanto lo scarto introdotto dalla compilazione casuale non riesce a colmare tutti i casi presentati dal manoscritto. Un timido tentativo che non esaurisce certo la flessibilità della scrittura manuale.

Alcuni accorgimenti sono stati adottati per superare i limiti della penna fissa di cui è dotato METAFONT: penne di diversa angolatura, effetti di curvatura dei tratti, *macro* in grado di simulare le grazie appena accennate che contraddistinguono lo scritto del *Codex Sinaiticus*.



(a) Immagine fotografica. (b) Contorno dei tratti. (c) Riempimento.

Figura 2: Composizione del glifo *alpha*, dal *Codex Sinaiticus* al *Simonides*.

5 Dal modello alla copia

Benché METAFONT sia uno strumento straordinario per il disegno dei contorni, esso mostra tutti i segni del tempo quando arriva il momento di assemblare in un unico font i singoli glifi così prodotti. METAFONT precede di molti anni l'introduzione delle diverse tecnologie che costituiscono l'attuale stato dell'arte in materia di caratteri digitali. Laddove METAFONT poteva gestire al massimo 256 glifi in un font, Unicode permette oggi di catalogare virtualmente ogni glifo, appartenente a un qualsiasi alfabeto, all'interno di un unico spazio di codifica, eliminando di fatto il ricorso a una pletora di codifiche, spesso *ad hoc*, che rendevano difficoltoso l'uso di un font su diverse piattaforme e in programmi diversi. Type1 e TrueType rappresentano i due standard alternativi per il disegno dei contorni. Il formato OpenType, combinando insieme Unicode, Type1/TrueType e una serie di strumenti che semplificano l'accesso a operazioni complesse quali sostituzioni di varianti, ecc., è ormai diventato la tecnologia di riferimento per i caratteri digitali.

Il problema di ottenere, a partire da sorgenti METAFONT, dei font in un formato corrispondente allo standard *de facto* del momento (ieri Type1, oggi OpenType) è stato già più volte affrontato e risolto in passato.[1] Il metodo più comune, oggi, consiste nel compilare i singoli glifi tramite METAPOST e poi usare FontForge come libreria per riassemblare i glifi in un unico font. Questa operazione è automatizzata tramite uno script Python che si incarica anche di aggiungere tutte le eventuali *feature* OpenType (nel nostro caso l'accesso a varianti casuali).

Un esempio di questo approccio è rappresentato da `mf2outline`,[2] un sistema messo a punto da Linus Romer inizialmente per la realizzazione di FetaMont. Il sistema consta essenzialmente di due parti: un insieme di macro che estendono METAFONT/METAPOST e permettono, per esempio, di usare codici Unicode per identificare i singoli glifi, di definire classi di crenatura o di specificare *lookup* per varianti randomizzate;[3] e uno script Python che gestisce l'intera compilazione.

Lo script è ben fatto, ma pensato per un font molto particolare (è un'estensione del font usato per il logo di METAFONT/METAPOST) e con un numero limitato di varianti. Nel caso del *Simonides* (che pure è un font molto particolare) si possono avere fino a 38 varianti di un singolo carattere. Si pongono quindi due problemi, uno di ordine concettuale e uno di ordine pratico.

Il primo problema concerne il modo di codificare queste varianti. La regola vorrebbe che alle varianti di un glifo (maiuscoletto, varianti stilistiche, varianti casuali) non venga assegnato un codice Unicode e che esse siano selezionabili solo attraverso la rispettiva *feature* OpenType. Questo non è possibile, nel nostro caso, perché lo script sfrutta esplicitamente il valore del codepoint Unicode durante la compilazione dei singoli glifi. Si è deciso, quindi, di ripiegare sulla soluzione più vicina in termini di accettabilità: alle varianti viene assegnato un codepoint che si trova all'interno della *Private User Area*. Ciò garantisce che non ci siano collisioni con caratteri che hanno una codifica Unicode, anche se non si possono escludere ambiguità in caso di sostituzione del font (font diversi possono usare la *PUA* per codificare glifi con diverso significato). A parte situazioni particolari, tuttavia, questo modo di procedere non dovrebbe creare grattacapi.

Il secondo problema consiste nell'evitare di dover riscrivere 38 volte, una per ogni variante, il codice di ciascun glifo. Una semplice macro, riportata nella figura 3, è sufficiente a questo proposito. La macro, `char_with_variants`, genera un glifo (ad esempio `alpha`) e tutte le sue varianti ordinarie. La macro richiede cinque argomenti, di cui l'ultimo (`body`) è il codice che disegna il glifo. I primi quattro argomenti rappresentano rispettivamente il codice Unicode del carattere, dato in forma di numero esadecimale; il codice, nella *PUA*, in cui

1. Alcuni tra i programmi sviluppati a questo scopo: `m2ps` (Yanai e Berry, 1990); `mftrace` (Nienhuys, 2017); MetaFog (Kinch, 1995); METATYPE1 (Jackowski *et al.*, 2003); MFLua (Scarso, 2011, 2012). Utili informazioni si trovano anche in Píška (2005).

2. `https://github.com/linusromer/mf2outline`; se ne trova una breve descrizione in Romer (2014).

3. Il metodo usato per specificare il *lookup* `randomize` può essere facilmente esteso anche ad altri *lookup* come è stato verificato durante i nostri esperimenti con il *Simonides*, anche se alla fine `randomize` è stata l'unica *feature* OpenType che abbiamo implementato.

```
% Variants
string codepoint; % Full unicode codepoint.
string variant;   % Encodes the symbolic name of each single variant in a block ("alpha.1", "alpha.2", ecc.).

def char_with_variants(expr code, start, num_var, name) (text body) =
  % draws a single character and its variants
  % code = unicode codepoint
  % start = position of the first variant; should be in the PUA
  % num_var = number of variants
  % name = conventional name
  % body = glyph code
  addrandvariant(code,code); % It is convenient to add here infos for the 'randomize' lookup
  variant:= "Unicode " & name;
  codepoint:= code;
  body;
  for i=1 step 1 until num_var: % Usually we have 32 variants
    codepoint:= hexadecimal(start+i-1);
    variant:= name & "." & decimal(i);
    body;
    addrandvariant(code,codepoint); % Add variant to 'randomize' lookup
  endfor;
enddef;
```

Figura 3: Il codice per generare automaticamente l'intero insieme delle varianti di un glifo.

cominciare a codificare le varianti casuali relative al glifo, sotto forma di numero decimale; il numero delle varianti; il nome del glifo. Purtroppo non è possibile esprimere il "punto di partenza" delle varianti casuali sotto forma di numero esadecimale, come è consueto per Unicode, a causa di una limitazione del comando `hexadecimal`, usato nella conversione da valore esadecimale a valore decimale, il quale, anche nel caso sia stata passata all'eseguibile `mpost` un'opzione per l'uso del calcolo in virgola mobile, restituisce un errore per valori che eccedono i limiti classici del programma METAFONT. La macro esegue una prima volta il codice contenuto in `body`, gli assegna il relativo codepoint e lo aggiunge a una tabella di sostituzioni (`addrandvariant(code,code)`). Poi ripete l'operazione, all'interno di un ciclo, tante volte quante sono le varianti volute, posizionando i glifi così ottenuti nella *PUA*, in slot adiacenti a partire dal valore `start` indicato, e aggiunge ciascuna variante alla tabella di sostituzioni di cui sopra. Per il carattere `alpha` il codice assume l'aspetto seguente:

```
char_with_variants("03B1",57344,32,"alpha",
beginsimonideschar(codepoint,6,cap_height#,0,1,2);
  variant;
  z1=pp(2hair_space,bot h);
  z2=pp(lft w-1/6hair_space,0);
  z0.5=z1+2serif_a;
  stroke1= z0.5...z1{x2-x1,1.3(y2-y1)}..z2;
  z3=point 1.1 of stroke1;
  z4=pp(rt 1/6hair_space,0);
  stroke2= z3{x4-x3,1.8(y4-y3)}..z4;
  z5=point .8 of stroke2;
  z6=point 1.5 of stroke1;
  stroke3= z5{dir pa}..z6;
  draw stroke1;
  pickup simonides_pen_tilted_b;
  draw subpath(0.02,length stroke2) of stroke2;
  draw subpath(0,.98) of stroke3;
  penlabels(0.5,1,2,3,4,5,6); endchar;
);
```

6 Sfogliare le pagine del *Codex Sinaiticus*

Non è facile testare un font come il *Simonides* senza collocarlo in un contesto adeguato, che consenta al tempo stesso di reggere il confronto con il manoscritto originale da cui è estrapolato. Si è scelta a questo scopo una pagina del Vangelo di Giovanni, che presenta delle caratteristiche alquanto peculiari.

Innanzitutto si deve tener presente la natura del *Simonides*, che è un font OpenType dotato di molte varianti. Per gestire questa complessità si è fatto ricorso a LuaLaTeX come motore di composizione, il quale si è dimostrato perfettamente in grado di gestire tale compito.

Cruciale, in questo senso, si rivela l'impiego del pacchetto fontspec, poiché permette un controllo accurato delle specifiche relative ai font OpenType. Consente altresì di impostare il parametro `rand` (Random), essenziale per la distribuzione automatica delle varianti di ogni glifo.

```
\usepackage{fontspec}
\setmainfont[Renderer=Full,
             Letters=Random,
             WordSpace=0,
             Opacity=0.9,
             Script=Greek]{simonides.otf}
```

Per quel che concerne la configurazione della pagina, non si riscontrano grandi difficoltà, pertanto la classe article risulta più che sufficiente, se corredata dei pacchetti geometry (per le dimensioni del foglio e del blocco del testo), fancyhdr (per le testatine) e luacolor (per gli effetti di colore).

Le pagine del *Sinaiticus* sono abbastanza ampie (380 mm×345 mm) e il testo viene distribuito prevalentemente su quattro colonne e di rado su due, ad esempio per i libri poetici del Vecchio Testamento.

Figura 4: Riproduzione di *Codex Sinaiticus*, BL 255: Giovanni 12, 6 sgg.

Per semplificare la costruzione di questa struttura si è ricorso a multicol, che risolve la maggior parte dei problemi. L'uso di questo pacchetto però ha rivelato almeno due limiti per i nostri scopi, la cui soluzione ha richiesto opportuni accorgimenti.

Ad esempio, al fine di ottenere un'armonica distribuzione delle righe che devono presentarsi equidistanti, si è pensato di aggiungere i seguenti comandi.

```
\baselineskip=1\baselineskip plus 1fill
\lineskip=0pt plus 1fill
\lineskiplimit=-\maxdimen
```

Un'ulteriore complicazione deriva dall'impossibilità del pacchetto di gestire adeguatamente le note a margine, che nel *Sinaiticus* si presentano al fianco sinistro di ogni colonna. Si tratta propriamente della notazione dei versetti biblici, risolta con l'indicazione di un comando in grado di creare una scatola di testo nello spazio voluto.

```
\newcommand{\versiculus}[1]{%
  \makebox[0pt][r]{%
    \parbox[t][][t]{10mm}{%
      \centering\baselineskip=2.6mm #1
    }
  }
}
```

Per concludere, è necessario considerare altri aspetti del testo. La versione riprodotta è quella fornita in formato XML dal sito ufficiale del progetto, nato per valorizzare lo studio e la divulgazione del manoscritto. Come già accennato, nella composizione digitale si è dovuto ricorrere a caratteri più minuti, che figurano abbastanza spesso al termine di ogni riga, probabilmente usati dallo scriba per giustificare quanto più possibile

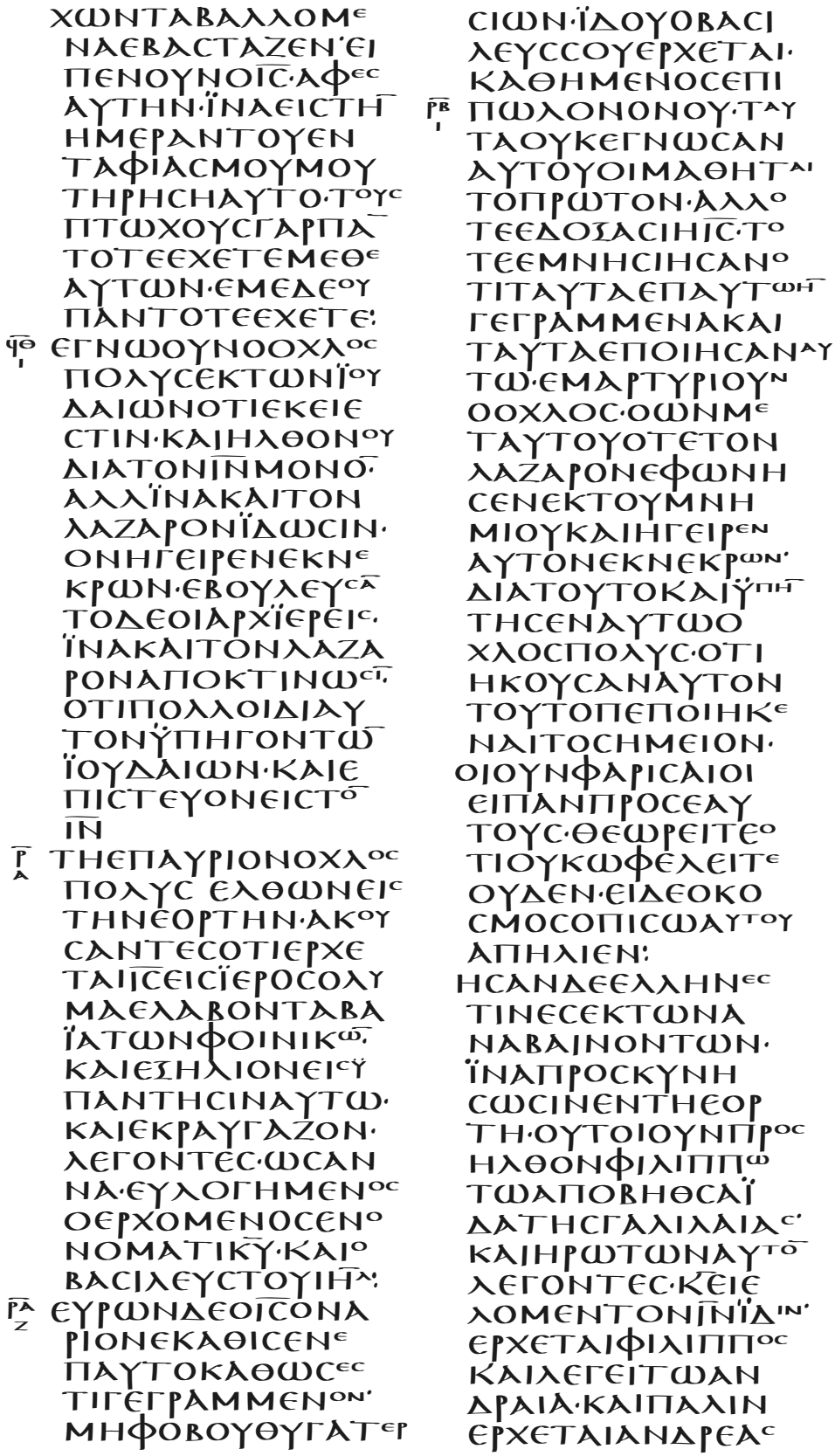


Figura 5: Riproduzione di *Codex Sinaiticus*, BL 255: Giovanni 12, 6 sgg. Dettaglio delle prime due colonne.

il margine della colonna. Il font *Simonides* è dotato di questi particolari caratteri, così come dei segni diacritici e di interpunzione che si possono incontrare nell'originale.

In definitiva, esistono sul mercato diversi font che riproducono l'alfabeto di un onciale greco, e altri comunque validi vengono forniti gratuitamente. Il confronto con il *Simonides* può essere solo parziale, perché diverse sono le modalità e le finalità di utilizzo. L'unico paragone pertinente di cui si ha conoscenza, relativo quindi a un carattere tipografico ispirato alla calligrafia del *Sinaiticus*, è quello in metallo utilizzato per l'edizione in facsimile del 1862 e curata dallo stesso Tischendorf. Questo si presenta molto bello, sufficientemente leggibile e al tempo stesso fedele: un contributo ancora essenziale per lo studioso d'oggi. Il *Simonides* ne vuole essere un ideale perfezionamento, come alternativa al facsimile fotografico Tischendorf (1862).

7 Interpretazioni

L'idea di un *copista digitale* non è nuova, ma qui trova una delle sue più effettive reincarnazioni.

Alcuni aspetti vanno però segnalati, che ripropongono il dilemma dell'interazione uomo-macchina. L'aleatorietà con cui vengono distribuiti i glifi sulla riga richiama al lettore il moto vivace della scrittura calligrafica. Ma il calcolatore obbedisce alle regole progettate, privo quindi della facoltà di effettuare scelte contestuali.

L'amanuense cerca di ritrovare, nel disordine del mutamento repentino, una regola condivisa. Nel tentativo di riprodurre il modello mentale della scrittura, introduce suo malgrado dei sottili e spesso involontari cambiamenti: l'esecuzione tradisce il riferimento ideale. La macchina, inevitabilmente rigida, viene forzata a inserire nello schema elementi di rottura della rigidità, ma è guidata dal caso e non da principi di adattamento: la variazione è irrazionale e non guidata dal gusto artistico.

Tuttavia la simulazione inganna anche l'occhio più esperto, e rende possibile la meraviglia del gioco di prestigio.

8 Ouverture

Constantinos Simonides (1820–1890) fu un paleografo colto e profondo conoscitore di antichi manoscritti, abilissimo e impareggiabile calligrafo, nonché uno dei più grandi falsari di tutti i tempi Schaper (2013). Sebbene smentito da eminenti studiosi, Tischendorf e Bradshaw soprattutto, egli si attribuì la paternità del *Codex Sinaiticus*, come frutto di puro apprendistato giovanile.

A lui si è pensato di dedicare il nostro lavoro.

Riferimenti bibliografici

Codex Sinaiticus Project (2015). «Codex Sinaiticus». URL http://www.codexsinaiticus.org/en/.

Hoekwater, T. e Hagen, H. (2008). «Punk from Metafont to MetaPost». *MAPS*, (37), pp. 55–58. URL http://www.ntg.nl/maps/37/10.pdf.

Jackowski, B., Nowacki, J. M. e Strzelczyk, P. (2003). «Programming PostScript Type 1 Fonts Using METATYPE1: Auditing, Enhancing, Creating». *TUGboat*, **24** (3), pp. 575–581. URL https://www.tug.org/TUGboat/tb24-3/jackowski.pdf.

Kinch, R. J. (1995). «MetaFog: Converting METAFONT Shapes to Contours». *TUGboat*, **16** (3), pp. 233–243. URL https://www.tug.org/TUGboat/tb16-3/tb48kinc.pdf.

Knight, S. (1998,2009). *Historical Scripts: From Classical Times to the Renaissance*. Oak Knoll Press.

Knuth, D. (1988). «A Punk Meta-Font». *TUGboat*, **9** (2), pp. 152–168. URL http://tug.org/TUGboat/Articles/tb09-2/tb21knut.pdf.

Nienhuys, H.-W. (2017). «mftrace». URL http://lilypond.org/mftrace/.

Parker, D. (2010). *Codex Sinaiticus: The Story of the World's Oldest Bible*. British Library.

Píška, K. (2005). «Converting METAFONT sources to outline fonts using METAPOST». *TUGboat*, **26** (2), pp. 158–164. URL https://www.tug.org/TUGboat/tb26-2/piska.pdf.

Romer, L. (2014). «Fetamont: An extended logo typeface». *TUGboat*, **35** (1), pp. 17–21. URL https://www.tug.org/TUGboat/tb35-1/tb109romer.pdf.

Scarso, L. (2011). «Mflua». *TUGboat*, **32** (2), pp. 177–184. URL https://www.tug.org/TUGboat/tb32-2/tb101scarso.pdf.

— (2012). «MFlua: Instrumentation of METAFONT with lua». *ArsTEXnica*, (14), pp. 72–81. URL https://www.guitex.org/home/images/ArsTeXnica/AT014/mflua.pdf.

Schaper, R. (2013). *L'odissea del falsario. Storia avventurosa di Costantino Simonidis*. Bononia University Press.

Tischendorf, K. (1862). *Bibliorum Codex Sinaiticus Petropolitanus*. Giesecke & Devrient. URL http://www.sbible.ru/sinaj2.htm.

Yanai, S. e Berry, D. M. (1990). «Environment for Translating METAFONT to POSTSCRIPT». *TUGboat*, **11** (4), pp. 525–541. URL https://www.tug.org/TUGboat/tb11-4/tb30yanai.pdf.

▷ Claudio Vincoletto
Torino
claudio dot vincoletto at gmail dot com

▷ Massimiliano Dominici
Pisa
mlgdominici at gmail dot com