UNIVERSIDADE D
COIMBRA

Hugo Thumann Mendes Vale Pereira

Entre o Desenho e a Escrita

Dissertação no âmbito do Mestrado em Design e Multimédia orientada por Ana Sabino, João Bicker e Evgheni Polisciuc e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Outubro de 2020

Entre o Desenho e a Escrita

Hugo Thumann Mendes Vale Pereira

Orientação
Ana Sabino
João Bicker
Evgheni Polisciuc

Mestrado em Design e Multimédia

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
da Universidade de Coimbra

4

Agradecimentos

A finalização do presente trabalho, em tempos tão diferentes dos que me tenho vindo a habituar, não teria sido possível sem o apoio e a resiliência de um conjunto de pessoas que me acompanharam durante o seu decorrer e às quais agradeço.

Em primeiro lugar, aos meus orientadores, a Ana Sabino, o Professor João Bicker e o Professor Evgheni Polisciuc, sem cujo o acompanhamento, paciência e conhecimento este projeto estaria ainda por começar.

À minha família, pela confusão, pelo apoio de sempre e pela confiança que depositam em mim.

À Renata, pelo constante apoio, pela diversão e pela crítica.

Aos meus amigos, pela motivação que oferecem.

E ao Plancton pela companhia em todas as longas noites de trabalho.

6

Resumo

Vivemos rodeados de desenhos que lemos. A leitura tornou-se algo tão natural que nos desprendemos da sua natureza pictórica. Olhamos para as letras como algo absoluto.

Em "A Reinvenção da Leitura: breve ensaio crítico seguido de 19 poemas visuais", Ana Hatherly traça uma história do texto-imagem. É objetivo desta tese interpretar a obra de Ana Hatherly e representar as suas ideias aproveitando e explorando a capacidade de processos computacionais.

Partindo da ideia de escrita assémica, e da possibilidade de desconstrução do alfabeto em elementos modulares que, isolados, se esvaziam de sentido, e, conjugados, tornam a ganhá-lo, o objetivo será o de trazer esta abordagem para a prática artística atual, construindo a possibilidade de criação de artefactos visuais de geração automática que se encontrem na fronteira entre o legível e o ilegível, entre o desenho e o texto utilizando ao invés da caligrafia, a tipografia. Desenvolvendo assim um projeto de design computacional com capacidade de perguntar "o que é a escrita?" e "o que é escrita?", não com o objetivo de responder absolutamente a estas perguntas, mas precisamente com o objetivo inverso de as marcar como perguntas sem resposta absoluta.

Esta dissertação descreve o processo de trabalho no decorrer de um ano, consistente na conceptualização de uma ferramenta computacional automatizada para a exploração da natureza da escrita, também como os problemas técnicos e soluções encontradas para os mesmos, e apresenta uma série de avanços técnicos e conceptuais relevantes para a produção final do artefacto sugerido.

8

Abstract

We live surrounded by drawings that can be read. Reading has become something so natural for most of us that we forgot its graphic nature. Letters are seen as something absolute.

In “A Reinvenção da Leitura: breve ensaio crítico seguido de 19 poemas visuais” (“The Reinvention of Reading: brief critical essay followed with 19 poems”), Ana Hatherly writes about the history of the "textual image". Is the goal of this thesis to understand this Ana Hatherly's essay and express its ideas using and exploring the potentials of computational processes.

Starting from the idea of assemic writing, and of the possibility of breaking the alphabet into modular elements that by themselves don't have any special meaning but together do. It was our goal to bring this approach to the current artistic practice, building the possibility for the creation of visual artifacts automatically generated that can be placed at the border between readable and unreadable, between text and drawing. This, using instead of calligraphy, typography. Developing a computational design project with the ability to ask "what is the act of writing?" and "what can be perceived as writing?", not to answer these questions absolutely, but precisely with the reverse goal of marking those questions as not really answerable.

This essay explains the work process during one year, describing the conceptualization of an automatic and computational tool for the exploration of the nature of writing, but also the technical problems and the solutions found for them. It presents several conceptual and technical improvements meaningful to the final production of the suggested artifact.

10

Palavras Chave

Escrita assémica;
Tipografia;
Texto-imagem

Keywords

Assemic Writing;
Typography;
Textual image

12

Índice

1. Agradecimentos **5**
2. Resumo **7**
3. Palavras Chave **11**
4. Índice **13**
5. Introdução **15**
6. Estado da Arte **22**
7. Plano de Trabalho e Implicações **36**
8. Objetivos e Métodos **38**
9. Desenvolvimento **43**
10. Conclusão e Trabalho Futuro **125**
11. Bibliografia **127**

14

Introdução

Partindo do texto de Ana Hatherly de 1975, *A Reinvenção da Leitura*, fez-se uma análise da história da relação entre texto e imagem, entre textos que se veem e imagens que se leem, entre o gesto que produz escrita e a escrita que é o testemunho do gesto. De seguida, será necessário passar à concretização de uma forma de criar artefactos que ficam entre o visual e o verbal, exponenciando as capacidades da máquina de produzir sentido.

Os desenhos de Ana Hatherly valorizam certos valores típicos da caligrafia: a linha contínua, a diferença entre caracteres da mesma letra, a inexistência de alinhamento.

Fazendo parte da proposta o artefacto ser criado de forma computacional, a caligrafia será substituída pelo método de escrita natural neste meio, a tipografia. Desta forma, o artefacto produzido deverá também valorizar características típicas da tipografia, como a repetição e a modularidade.

Letra como ícone. Escrita como gravador de som

Antes de pensar na forma da letra interessa pensá-la pela sua função, e pela forma como a concretiza.

O objetivo final de uma palavra é ser lida. Existe como meio de transmissão de informação, comunicação. Exige um compromisso entre o escritor e o leitor, uma concordância. Permite que esse compromisso exista de uma forma gráfica. Para isso, é necessário que o leitor conheça o sistema que está a ser utilizado para transmitir awww informação. Caso contrário a escrita serve apenas como memorando de que quem a escreve. Como funciona então este protocolo? Em que é que está baseado? Porque é que o utilizamos? Porque escrevemos "árvore" ao invés de desenharmos uma? Existe algum motivo para o desenho de um se assemelhar à escrita do outro?

Ana Hatherly inicia o seu ensaio escrevendo "É preciso não esquecermos que a escrita alfabética é relativamente recente e que muito antes dela já se estabelecia a comunicação por imagens. Assim, se quisermos estudar a origem da poesia como escrita dum texto, nunca a poderemos dissociar do seu aspecto pictórico. Percorrendo a história mundial das imagens produzidas pelo homem, encontraremos quase sempre paralelamente escrita e imagem, sendo muitas vezes uma a outra." (Hatherly, 1975).

No mundo ocidental o sistema mais utilizado é a escrita fonética (baseada em fonemas), na qual os sons produzidos pela fala são atribuídos a símbolos (letras). Este sistema pressupõe o conhecimento de um

idioma falado que por sua vez poderá ser representado por letras. Assim, para a criação da escrita ocidental, a fala teve primeiramente de ser fragmentada. Foi necessária a invenção do fonema como parte de um todo, que neste caso é a palavra.

O mesmo não acontece em países orientais como a China, em que o sistema de escrita é logográfico. Desta forma o que é representado graficamente é diretamente a ideia do que se quer transmitir. Isto permite que dois falantes de dialetos distintos partilhem o mesmo sistema de escrita.

Experimentações: poesia concreta e escrita assémica

A própria letra é também ela desenhada, o desenho que cada letra tem é obviamente influenciador da percepção que o leitor tem acerca da palavra escrita. E existindo infinitas formas diferentes de utilizar e combinar as letras com o objetivo de descrever da melhor forma possível algo, deve também ser explorada a forma como essas mesmas letras podem ser desenhadas.

Se regras gramaticais podem ser quebradas de forma a procurar descrever algo de uma forma mais exacta. Também na forma como a palavra é desenhada essa experimentação deve ser feita.

Destas ideias surgem a poesia concreta e escrita assémica (Hatherly, 1975).

A poesia concreta parte do pressuposto, também assumido aqui, que a palavra não deve apenas ser lida verbalmente. Ela tem uma presença – física, visual, concreta – que faz parte do seu sistema e capacidade de criar sentido.

A escrita assémica vem da mesma maneira chamar a atenção para a visualidade da escrita, esvaziando-a de sentido, mas mantendo a sua manifestação gráfica enquanto escrita. São letras e conjuntos de letras que não necessariamente *significam*.

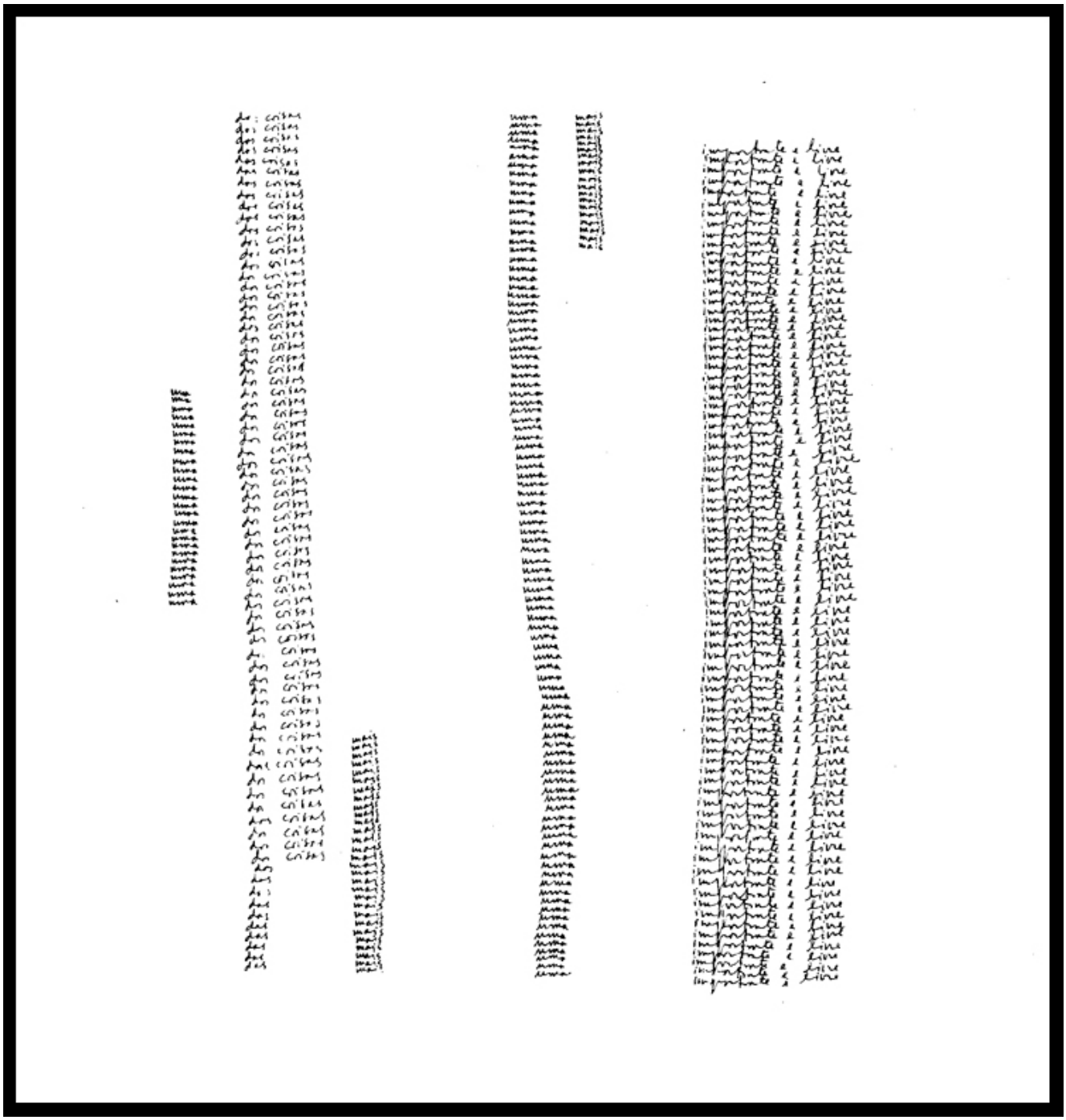
O que significa uma letra por si própria?

Ela é apenas a conjugação de um conjunto de letras que é capaz de criar sentido. Uma letra isolada não transmite sentido.

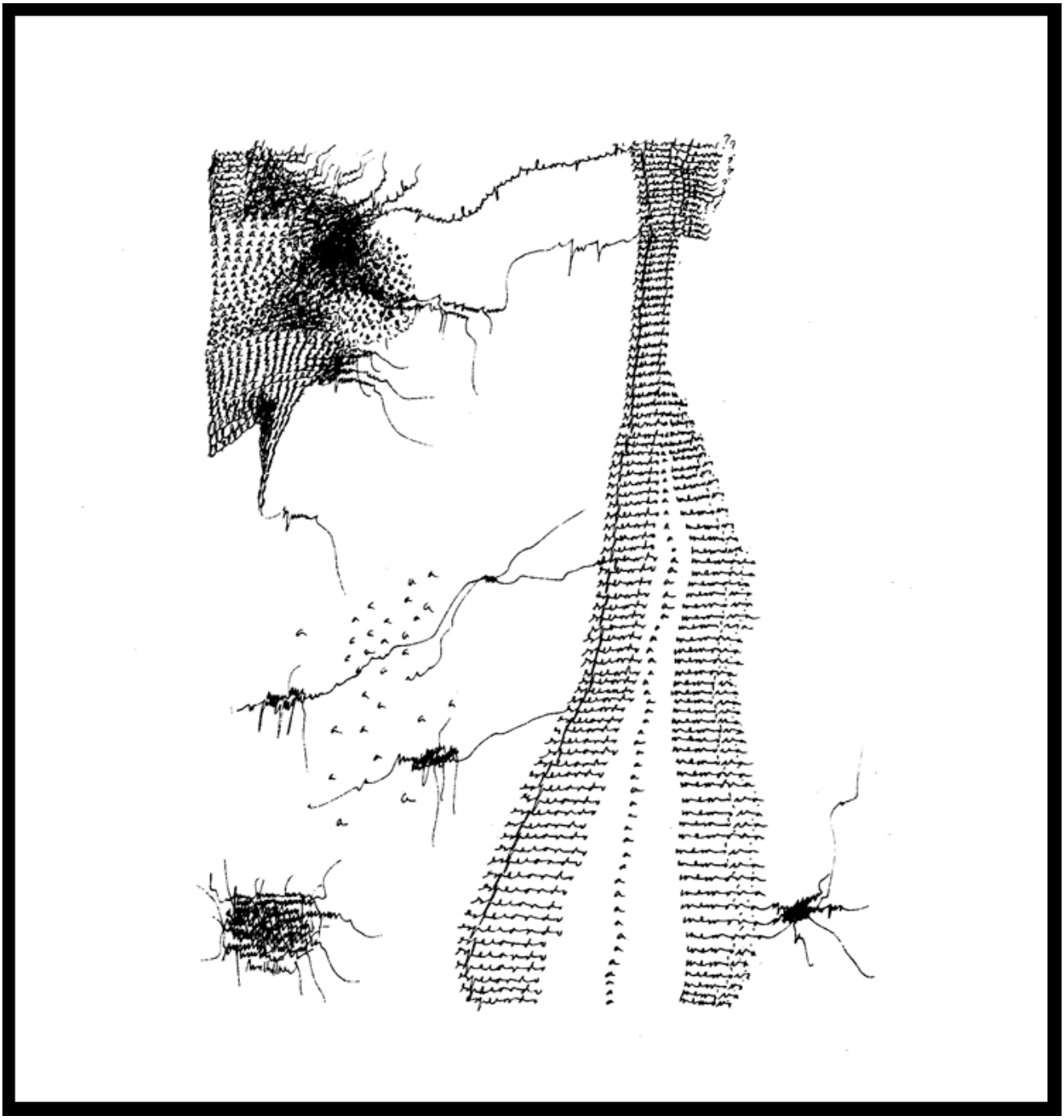
Por sua vez, uma letra é um conjunto definido de traços. Juntos, produzem uma letra. Esses traços isolados não são capazes de gerar sentido.

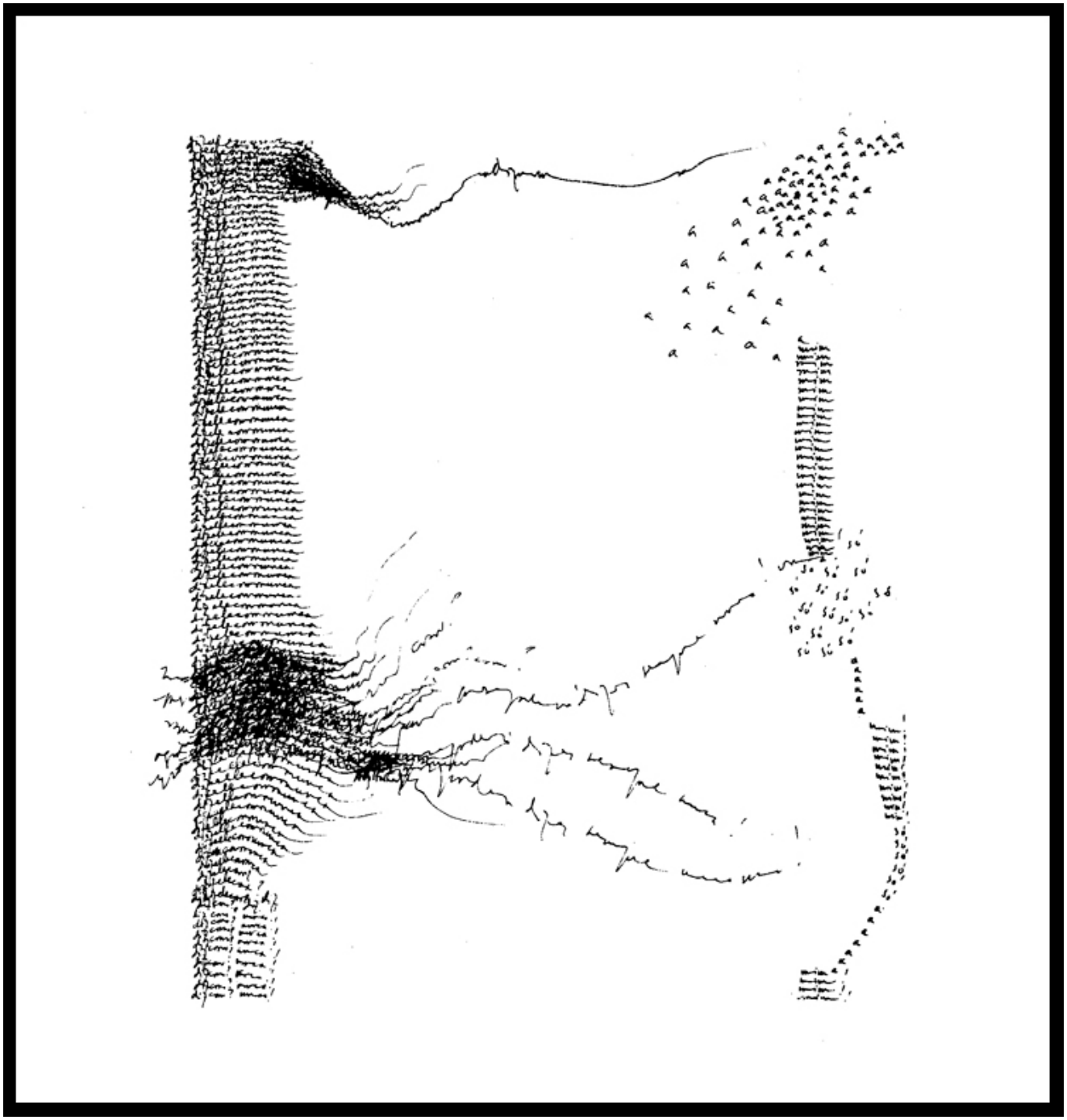
No entanto, dada a ubiquidade da letra tipográfica nas nossas vidas, somos capazes de as reconhecer mesmo quando não as conseguimos ler, o que aliás acontece quando somos confrontados com alfabetos que não lemos, como o árabe ou o cirílico.

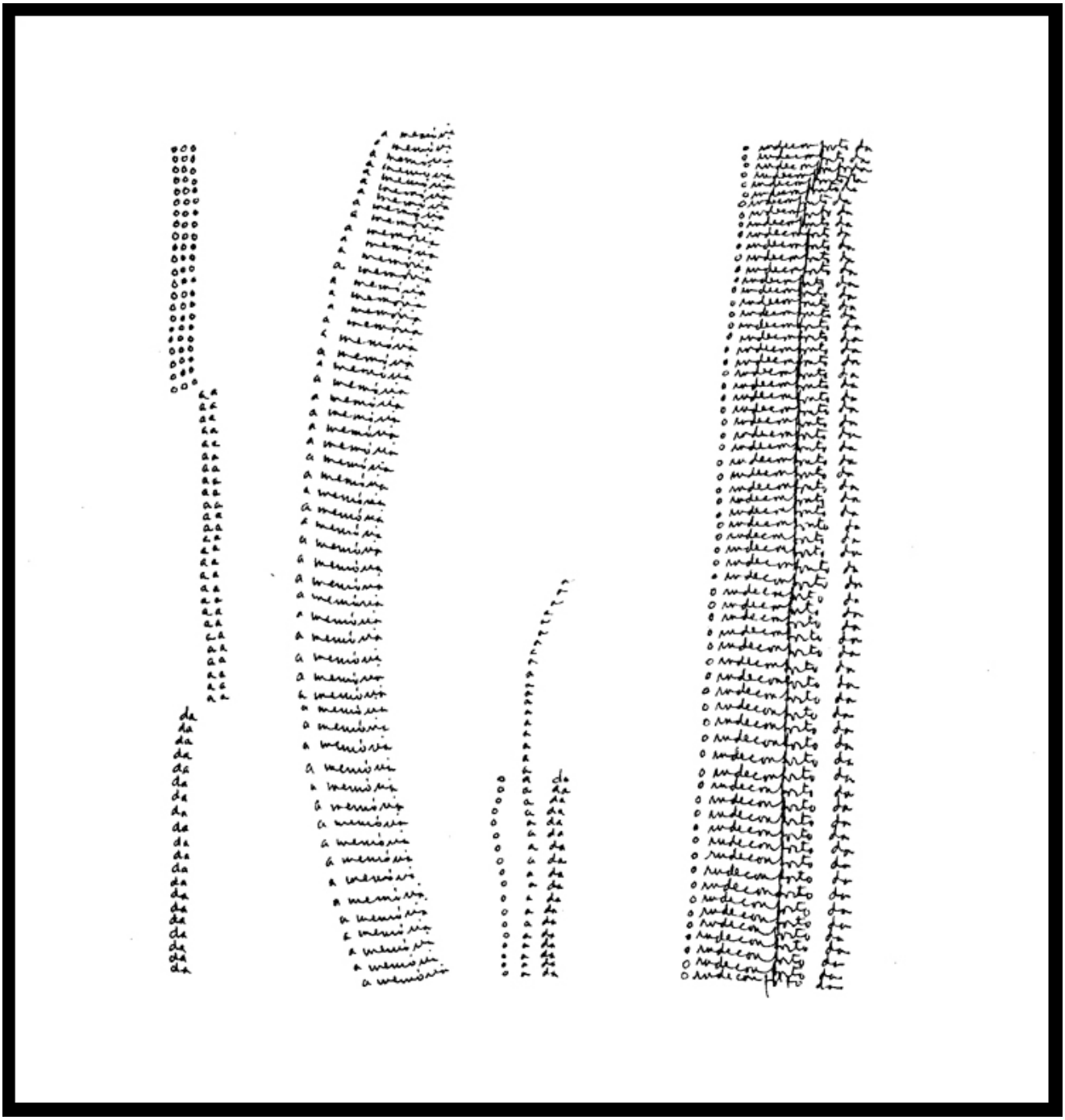
img 1 a 5 (pág 15 a 19)
5 textos visuais de "A Reinvenção da Leitura",
Ana Hatherly

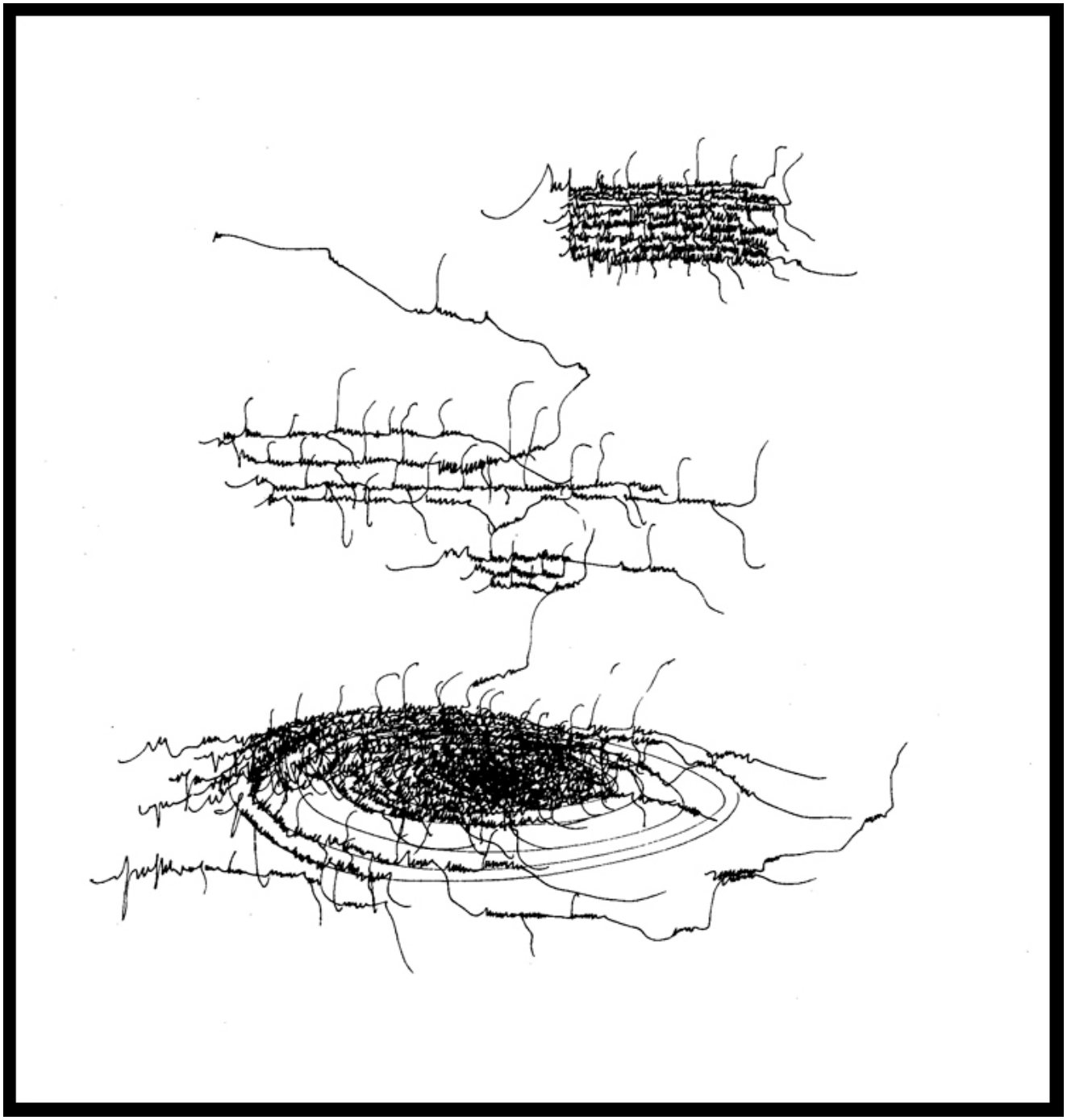


18









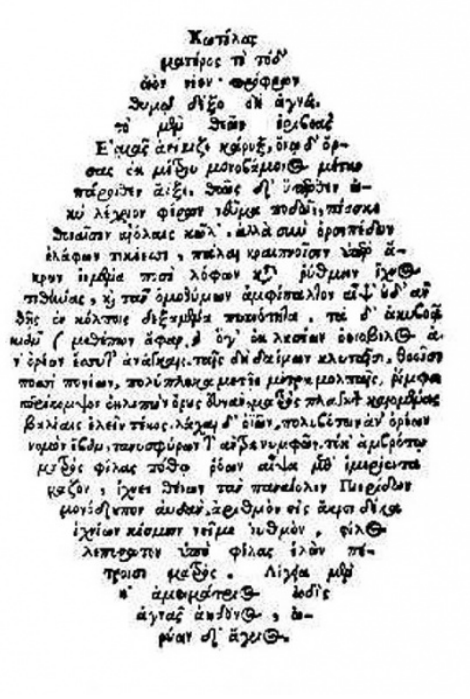
Estado da Arte

Contexto histórico

No ensaio "A Reinvenção da Leitura", Ana Hatherly apresenta inúmeros exemplos da história do texto-imagem. Sendo este texto o ponto de partida para o desenvolvimento da dissertação faz sentido analisar e referenciar alguns destes artefactos.

"Ovo", Símias de Rodes, 300 a. C.

É um poema que apresenta uma forma oval composta por linhas de texto. No entanto exige regras de leitura específicas como diz Ana Hatherly :"deve começar-se pela primeira linha superior, saltando depois para a última linha inferior, seguidamente retomando a segunda linha superior para descer à segunda inferior e assim sucessivamente até se atingir o centro" (Hatherly, 1975).



img 6
"Ovo", Símias de Rodes

Tratam-se de poemas que representam texto e imagem no mesmo espaço, Ana Hatherly referência os nomes de Porfyrius Optatianus, Alcuino e Bonifácio como alguns dos autores destes textos-imagem. É também referido que são poemas acrósticos, podendo ser lidos verticalmente e horizontalmente (Hatherly, 1975).

img 7
Carmina Figurata

Quais são os limites da tipografia?

Tem este capítulo, o objetivo de analisar a presença da tipografia no grupo de trabalhos selecionados. A sua seleção foi feita procurando características que revelassem alguma ambiguidade no que toca à sua natureza tipográfica.

Não pretendo com este capítulo procurar e definir uma fronteira onde termina o que é chamado de tipografia, mas sim alertar para a natureza espectral desta ferramenta. Tenciono apenas apresentar os motivos pelos quais algo pode ser considerado tipografia, ou pelo contrário, não o ser. Demonstrando que esta separação poderá ser mais complexa do que o esperado, sendo possível apresentar argumentos contraditórios.

Autómato “the Writer”
- Pierre Jaquet-Droz, 1772 (ref. 1)

Pierre Jaquet-Droz foi um relojoeiro suíço que viveu no século XVI. Para

além dos seus relógios, três das suas invenções mais interessantes são os autómatos humanóides “the Writer”, “the Musician” e “the Draftsman”. Tratam-se de três máquinas capazes de escrever, tocar música e desenhar. Neste contexto é interessante focarmo-nos no “the Writer”.

img 8 (topo)
Texto escrito pelo autómato

img 9 (baixo)
Sistema mecânico de programação do autómato

O autómato Escrivão, o mais complexo dos três autómatos, é uma máquina programável composta por 6000 peças, que escreve com uma pena.

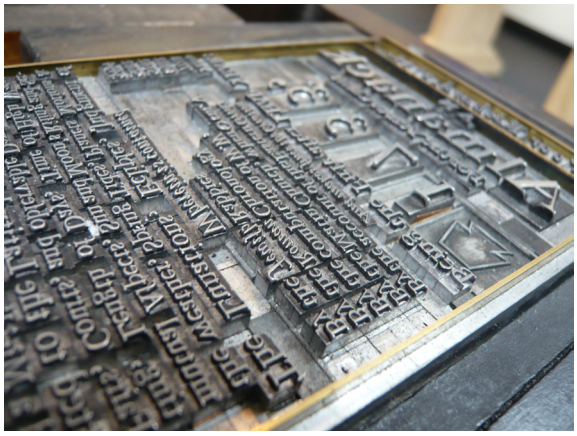
O autómato pode ser programado para escrever uma sequência de até 40 caracteres, formando palavras e frases, através de uma pena, o que nos remete imediatamente para a ideia de caligrafia. No entanto as letras escritas pelo autómato estão todas separadas, e o seu desenho, embora não pareça, já existe previamente. Ao contrário do usual em caligrafia, todas as letras "a" são representadas pelo mesmo glifo.

Aliás, o funcionamento do autómato assemelha-se bastante ao funcionamento de um software de gestão de fontes. O desenho da letra existe previamente, é guardado segundo um protocolo e, sempre que é necessário o tipo é aplicado num documento.

Pode então dizer-se que o autómato é na realidade uma máquina tipográfica.

Music Type (ref. 2)

Os tipos móveis inventados por *Gutenberg*, são fundidos em chumbo, e depois dispostos manualmente lado a lado (Meggs, 1983), de forma a compor palavras, frases e textos. Da mesma forma estes tipos musicais utilizam o mesmo princípio, mas com a finalidade de representar notas, melodias e acordes.



img 10 (topo)
Tipos móveis convencionais

img 11 (baixo)
Tipos móveis musicais

Ambos são utilizados para representar formas gráficas que representem som, tendo como únicas diferença o conjunto de glifos que utilizam, e o tipo de leitura utilizado. O som representado por um é o da palavra falada. O som representado pelo outro é o das notas musicais tocadas por um instrumento.

Pode-se falar de tipografia neste caso?

FF Scala Hands
- Martin Majoor, 1991 (ref. 3)

Bryan Ferry, RayGun - Artigo
- David Carson (ref.4)

No contexto mais usual, pensa-se na tipografia como a disciplina responsável pela disposição de letras e sinais de pontuação num suporte gráfico. No entanto ao analisar os glifos de uma fonte, reconhecemos por vezes caracteres que normalmente veríamos em contextos não tipográficos.

Exemplo disto é o caracter designado em inglês por *fist* ou *index* que representa uma mão que aponta com o dedo indicador. Embora se assemelhe mais a uma pequena ilustração que a uma letra, o *fist* é uma herança da caligrafia. Servindo para marcar parágrafos ou passagens importantes do texto este caracter é encontrado em impressões que datam 1484 (Sherman, 2005).

Um exemplo notável da utilização do *fist* é na fonte FF Scala, desenhada por Martin Majoor. A característica mais destacável desta família tipográfica é a da utilização de um mesmo esqueleto para o desenho de uma fonte serifada e uma não serifada, isto permite uma maior coerência gráfica quanto são utilizadas os dois tipos de fonte. Seguindo este princípio, também os *fist* das diferentes fontes foram desenhados de acordo com o estilo de cada uma respectivamente. Nas pequenas mãos da fonte serifada existe uma manga com terminações pontiagudas (aproximando-se das serifas), e nas da fonte não serifada as mangas terminam de forma aproximadamente rectangular (tal como numa fonte não serifada).

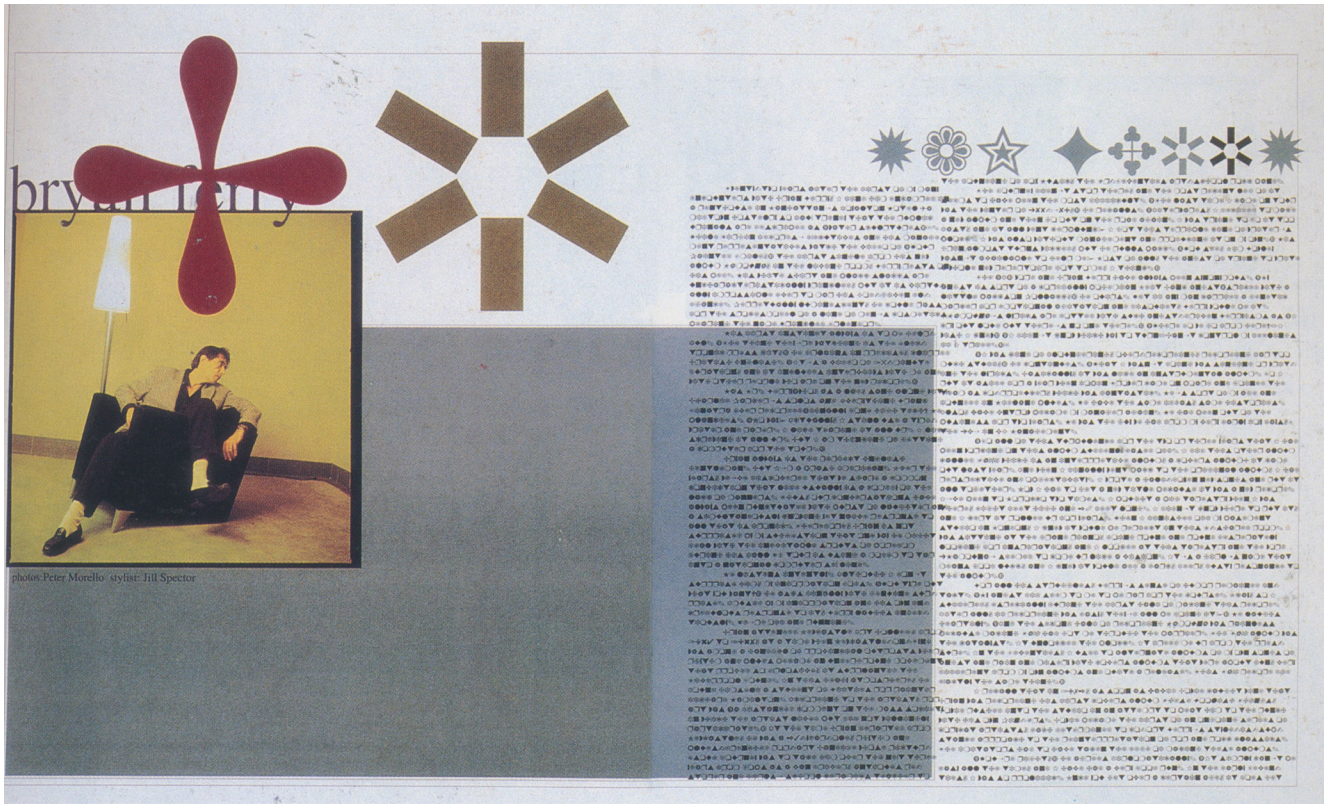
Tendo uma história registada no contexto tipográfico, e, como foi demonstrado a cima, existindo por parte dos *type designers* necessidade de tratar este glifo da mesma forma que os restantes, adaptando o seu desenho ao dos restantes numa fonte, penso que se pode dizer sem dúvidas que um pequeno desenho de uma mão pode ser um caracter tipográfico.

img 12
Glifos "mãos" da fonte Scala Sans
e Scala Serif

Mas, e se este caracter for disposto individualmente sem contexto tipográfico ao seu redor? Continua a tratar-se de tipografia? Ou será agora apenas um desenho de uma mão? E, neste contexto, será um glifo de uma letra, se disposto sozinho, também ele um caracter tipográfico? Uma consoante sozinha deixa de se ler de forma habitual. Passamos a vê-la como um desenho e não como uma letra. Uma letra "H", por exemplo, ao vermos a sua forma sem contexto pensamos no seu nome, e não no seu som (que é nulo na língua portuguesa, a não ser se precedido por um "C", "N" ou "L"). Se pretendêssemos representar o seu nome de forma a ser lido não deveríamos antes escrever "AGÁ?

O que diferencia os caracteres tipográficos dos não tipográficos? Outro exemplo poderá ser o da famosa composição gráfica feita por David Carson, na revista RayGun, para um artigo sobre o músico Bryan Ferry. Entediado com o artigo, Carson decidiu escolher uma fonte que o representasse, Zapf Dingbats. Nesta fonte o desenho de cada letra foi substituído por ícones sem significado aparente. No entanto trata-se de uma fonte real, cada letra tem um desenho único que se repete, os desenhos estão guardados num ficheiro de fonte e funciona como uma. O texto poderia ser "traduzido" para outra fonte, substituindo cada glifo um a um pelo respetivo na nova fonte, e lido normalmente. Ou, seria até possível o leitor aprender a ler Zapf Dingbats para poder ler o artigo. Seria um processo similar a aprender, por exemplo Braile. A informação está na mesma disponível no mesmo suporte, simplesmente o seu acesso está dificultado para a maior parte dos leitores por estes não estarem familiarizados com estes desenhos. Mas no entanto o texto ganha uma nova camada de interpretação. Citando Ellen Lupton, "Text is what language looks like".

O que define o que é ou não tipografia é o contexto em que algo é observado. Individualmente as próprias letras são desenhos, desprovidas de significado.



img 13
Artigo sobre Bryan Ferry, na revista RayGun

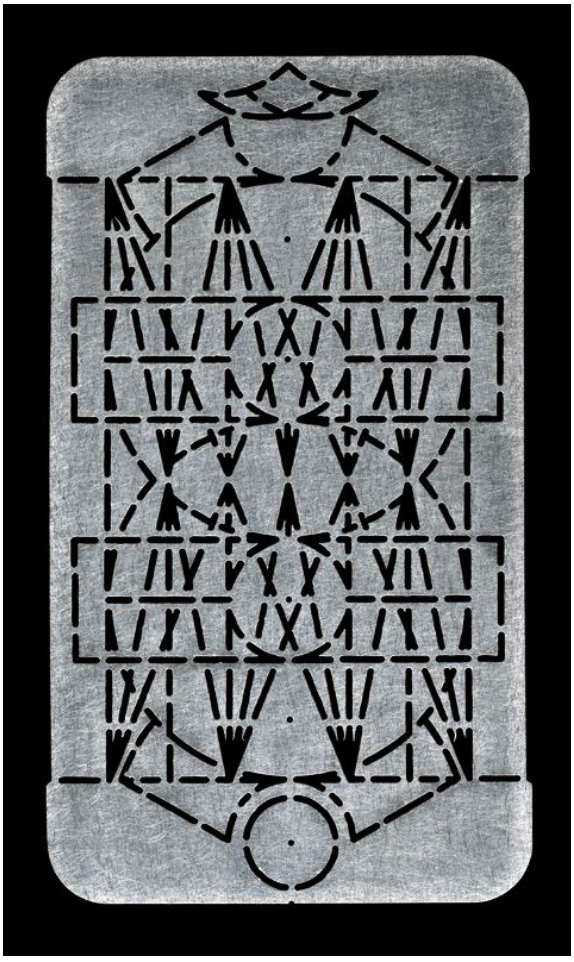
Fragmentação da letra

São as características mais antigas da tipografia que primeiro a identificam. Características que são herdadas dos artefactos utilizados na sua produção, como o tipo de chumbo, que impõe o espaçamento entre as letras; as caixas onde os tipos eram guardados, que batizam as maiúsculas e minúsculas; ou a repetição do desenho que representa cada letra de cada vez que esta aparece no texto. É por isto natural que estas características sejam assumidas como obrigatórias em tipografia. No entanto, da mesma forma que para a criação de fontes em chumbo, a palavra escrita caligraficamente teve ser separada nas suas partes, outro tipo de aplicações gráficas sugerem que a própria letra também pode ser separada nas suas partes.

PDU - Plaque Découpée Universelle - Joseph A. David (ref. 5 & 6)

Criada em 1876 por Joseph A. David, a *PDU* ou *Plaque Découpée Universelle* trata-se de um escantilhão que combina os traços necessários para desenhar qualquer letra do alfabeto. Neste caso a tipografia torna-se mais plástica, permitindo que o designer escolha a forma de cada, mas estando

restringido pelas curvas do escantilhão que mantêm a coerência entre os glifos. Versões digitais de fontes produzidas pela PDU foram criadas e distribuídas pela fundição Colophon Foundry, disponibilizando conjuntos de letras, maiúsculas e minúsculas e pontuação, mas também padrões ornamentais também produzidos a partir do escantilhão e alternativas estilísticas para cada glifo.



img 15
PDU - Plaque Découpée Universelle

Mecano Typeface
- Designer desconhecido, 1920 (ref. 7)

Fregio Mecano é uma fonte de origem italiana cujo autor é desconhecido. A sua produção parte também da ideia de que a letra pode ser seccionada nas suas partes que poderão por sua vez ser partilhadas entre várias letras. Trata-se então de um conjunto de peças, sem leitura tipográfica à partida mas que representam diferentes partes anatómicas da letra, que por sua vez podem ser justapostas para criar um conjunto infinito de glifos.

img 16
Fregio Mecano typeface

Interessa tentar localizar a tipografia nestes casos. O que é a fonte? Nestes exemplos, estamos perante uma infinitude de letras? Ou perante uma letra com infinitas formas?

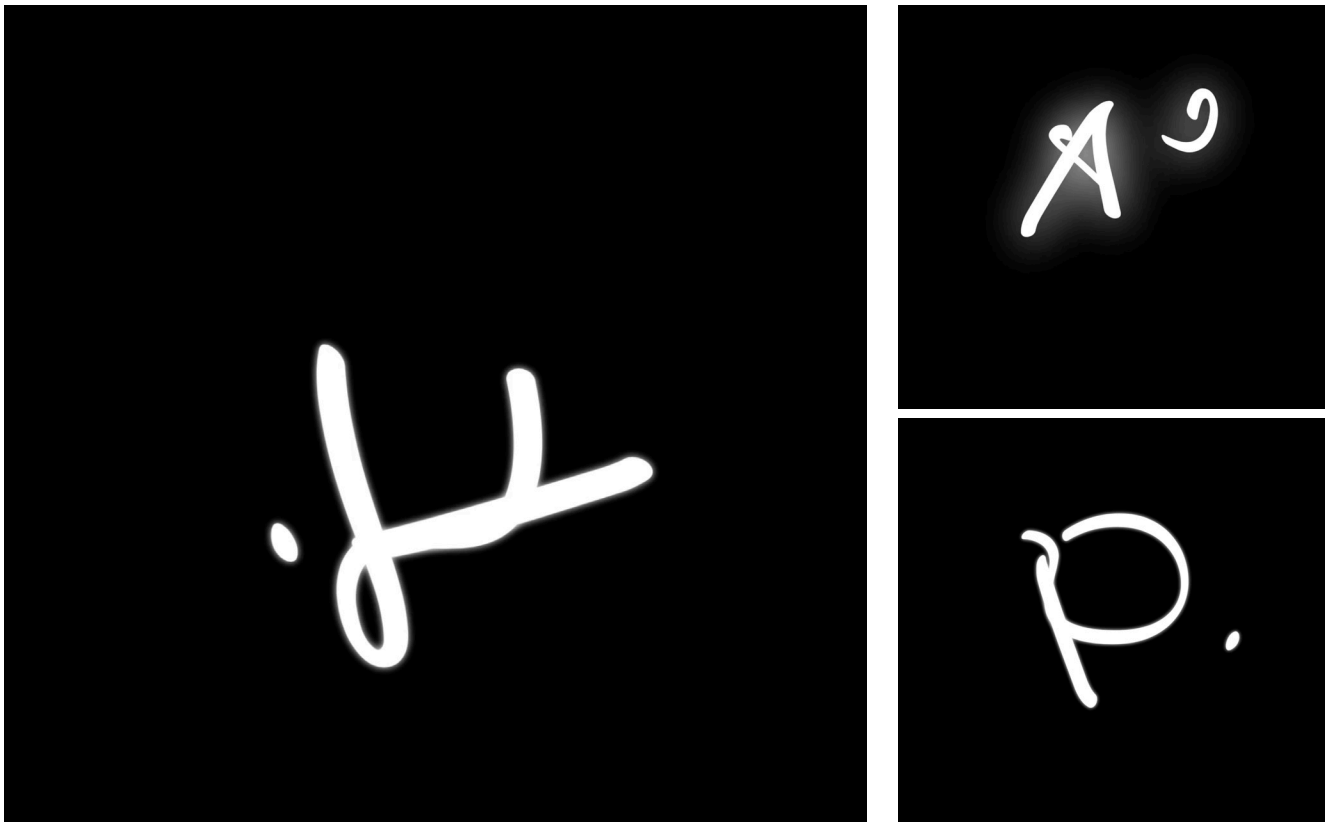
Casos de estudo atuais

Grammato
- Underware Type Foundry - 2019 (ref.8)

A fundição Underware Type Foundry, apresentou uma terceira abordagem de escrita, à qual chamam *Grammatography,* cujo objetivo é o de juntar características herdadas quer da caligrafia, quer da tipografia de forma a criar um sistema de escrita fluído (como o caligráfico) mas que seja aplicável digitalmente (como em tipografia).

Com esta a abordagem a letra não se comporta da mesma forma que as compostas tipograficamente, visto que a sua forma apresenta variações ao longo do tempo, um pouco como na caligrafia, em que o desenho de cada letra só está definido quando uma letra é efectivamente desenhada num suporte. No entanto no meio digital é possível que o movimento da letra não seja apenas na direção da sua escrita, mas também contrário a este, fazendo a letra desaparecer; ou até modificá-la apenas.

Isto levanta algumas das questões já propostas anteriormente. Se a letra pode modificar a sua forma até ao ponto de ser reconhecida como um outro caracter, poderá dizer-se que estamos na presença de apenas uma letra?



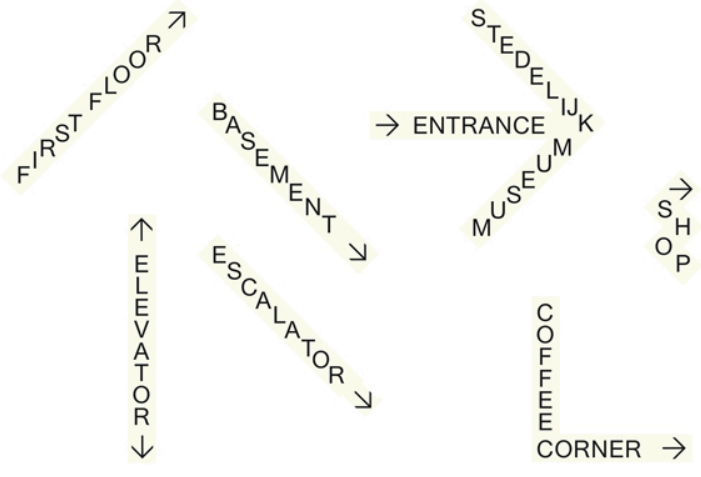
img 17
Grammato - Imagens de um mesmo glifo ao longo do tempo

Identidade do Stedelijk Museum Amsterdam
- Mevis & Van Deursen, 2012 (ref. 9, 10, 11 & 12)

Stedelijk Museum Amsterdam: Type/Dynamics Exhibition
- LUST & JURRIAAN SCHROFER, 2013 - 2014 (ref. 13)

Scraper Type
- LUST, 2006 - 2008 (ref. 14)

A identidade do *Stedelijk Museum Amsterdam* foi redesenhada em 2012 pelo estúdio de design Holandês *Mevis & Van Deursen.* O logotipo criado trata-se de um caracter de um "S" maiúsculo composto com as letras das palavras do nome do museu "*Stedelijk Museum Amsterdam*". Esta abordagem foi muito criticada e discutida pela comunidade de designers e acusado de ser pouco legível. É importante recordar que a identidade do museu foi durante 10 anos trabalhada pelo famoso designer modernista Wim Crouwel e o seu logotipo era composto apenas pelas letras "SM" em Helvetica.

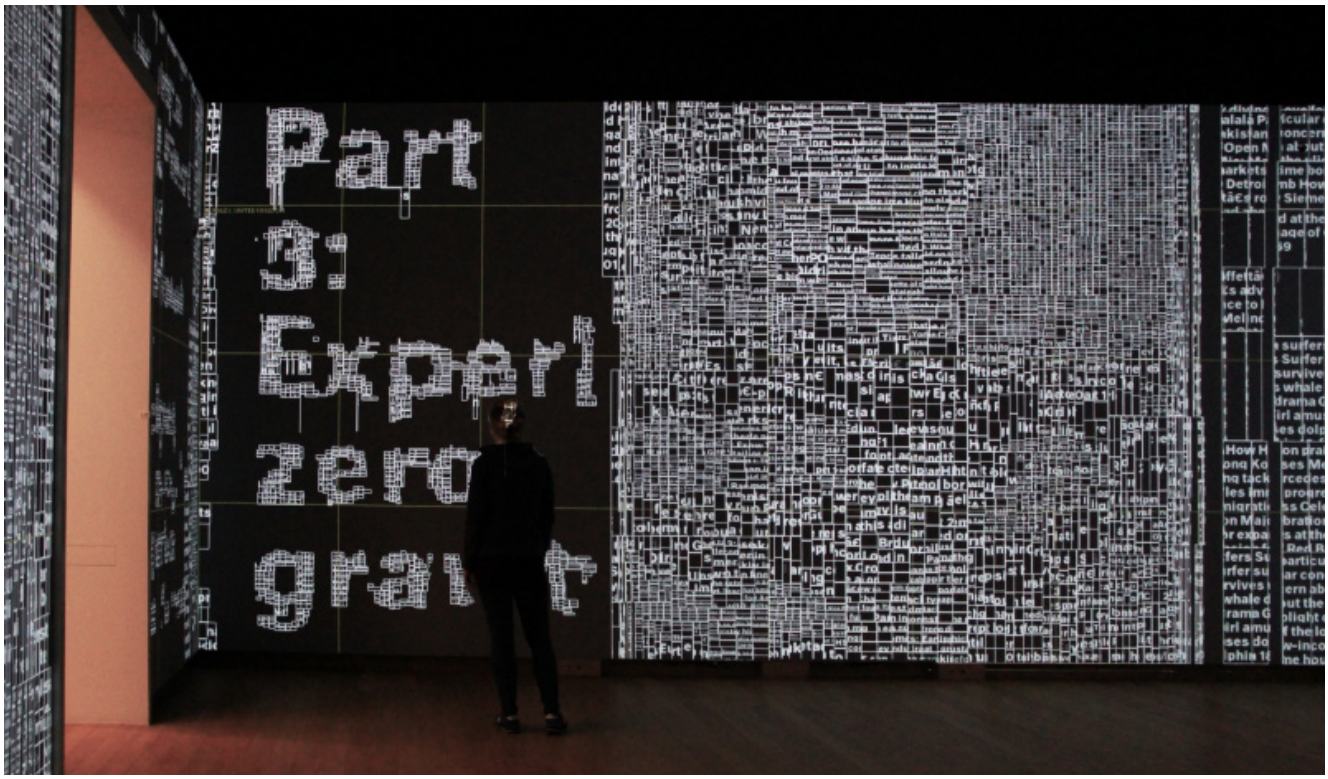


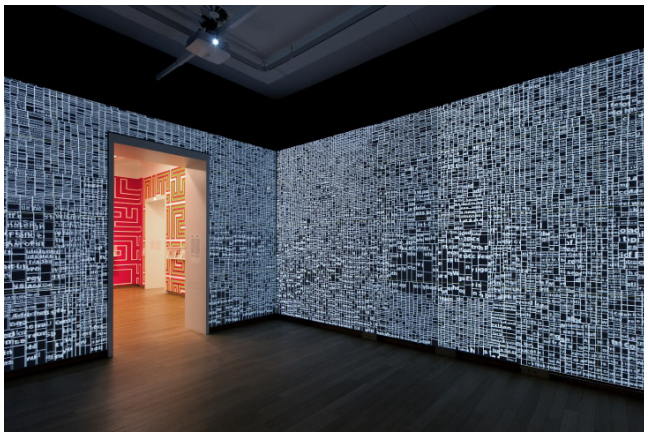
img 18 (topo)
Logo Stedelijk Museum

img 19 (baixo)
Aplicações da imagem gráfica

No contexto do nosso projeto interessa-nos a dimensão recursiva que a tipografia apresenta neste caso. Isto transforma a letra numa unidade atómica que se fôr observada de perto pode oferecer novas leituras também estas compostas por letras.

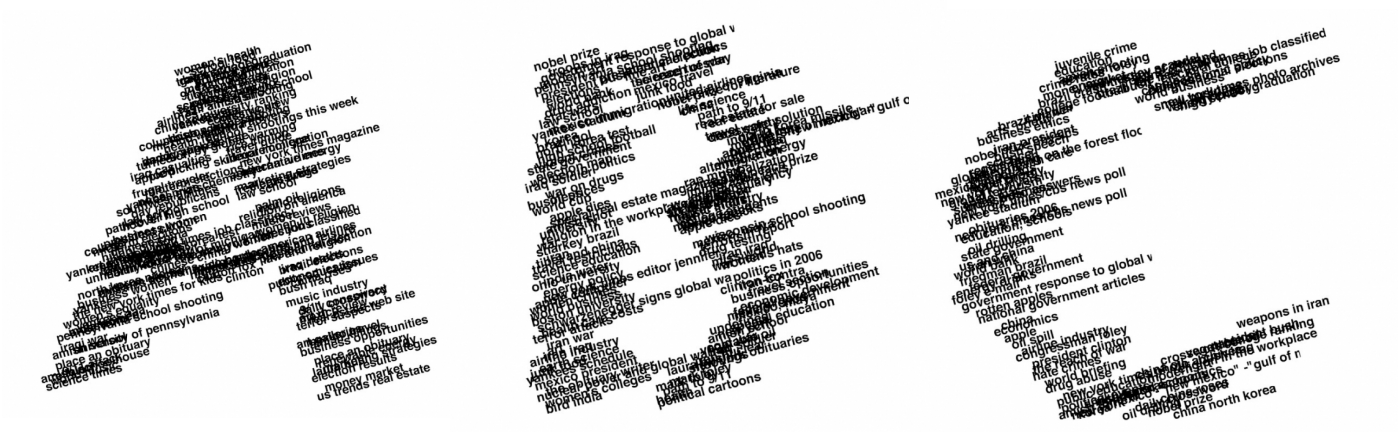
Um pensamento similar é visível na instalação *Type/Dynamics* do estúdio de design LUST que foi exibida também no museu Stedelijk em Amesterdão. Nesta instalação, imagens panorâmicas retiradas do Google Street View são representadas através de caixas de texto (LUST, 2013-14). Estas caixas de texto também elas se agrupam de forma a criar letras, criando dois níveis de leitura possível, o conteúdo da caixa de texto, e a letra formada por ela em conjunto com outras.

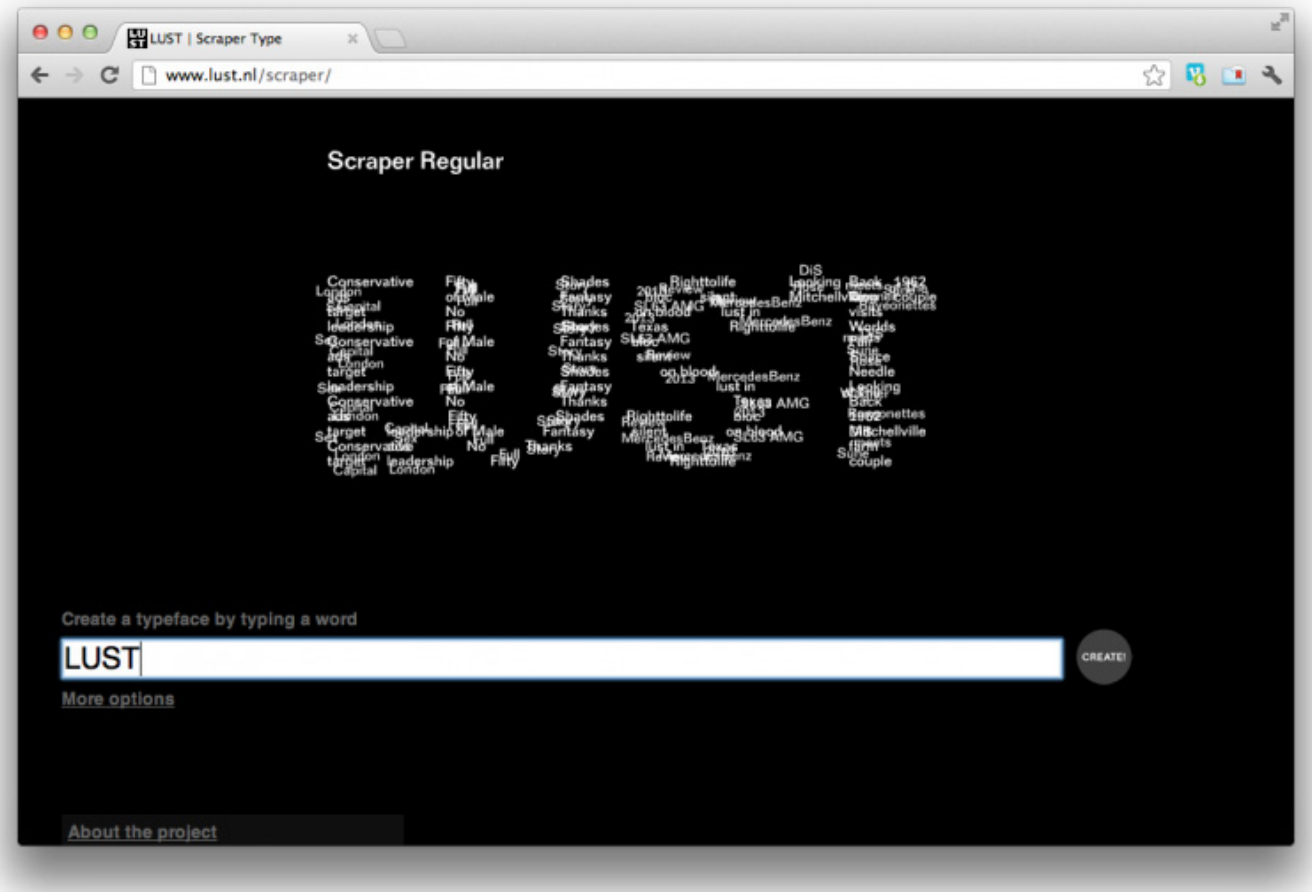




img 20 (topo, baixo esq. e dir.)
Instalação Type/Dynamics

Também no projeto *Scrapper Type* o estúdio LUST volta a utilizar a mesma lógica. Trata-se de uma aplicação que gera tipografia com base num *input* por parte do utilizador. No entanto, as letras são constituídas por palavras relacionadas com o tema escrito pelo utilizador (LUST, 2006-2008).



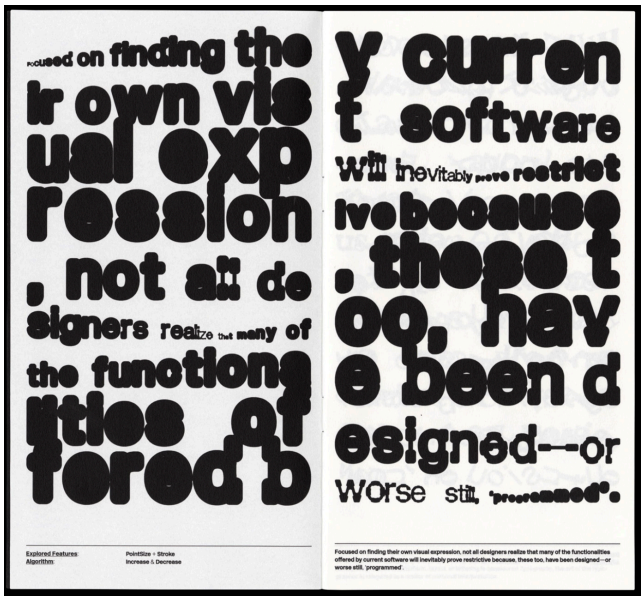


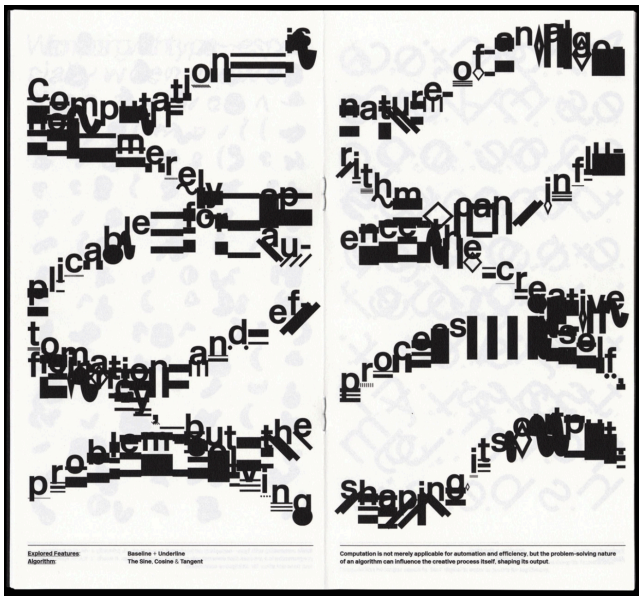
img 21 (topo & baixo)
Aplicação Scrapper Type

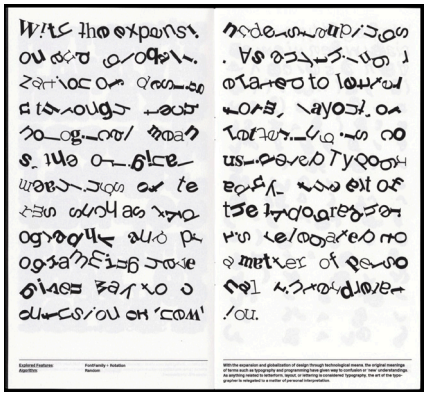
Scripting Typography
- Pedro Neves - 2018 (ref. 15 & 16)

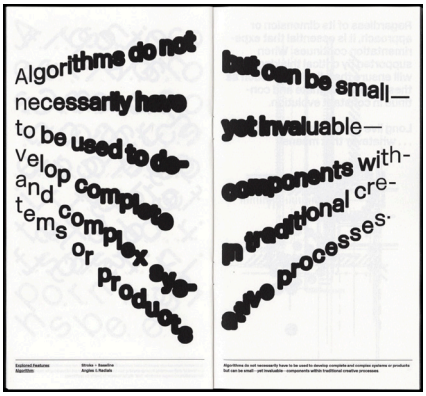
A tese de mestrado de Pedro Neves feita na *Basel School of Design* apresenta um estudo dos resultados que podem ser obtidos aplicando técnicas de *scripting* a tipografia. Neste caso a tecnologia utilizada é a biblioteca de *javascript "basil.js",* também desenvolvida na mesma escola, que permite a manipulação e automação de ficheiros Adobe InDesign. (Pedro Neves, 2018).

Estas experiências resultam em composições gráficas nas quais a letra é manipulada e distorcida, muitas vez deixando de ser reconhecida.





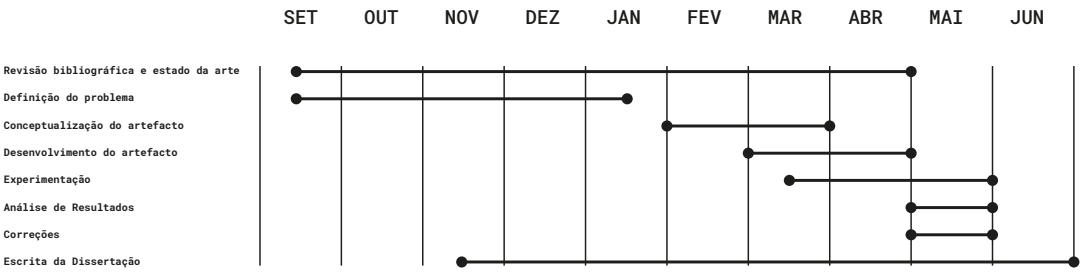




img 22 (todas)
Páginas de *Scripting Typography*, Pedro Neves

Plano de Trabalho e Implicações

O diagrama de Gantt a baixo representa a previsão inicial da distribuição das tarefas ao longo do ano lectivo.



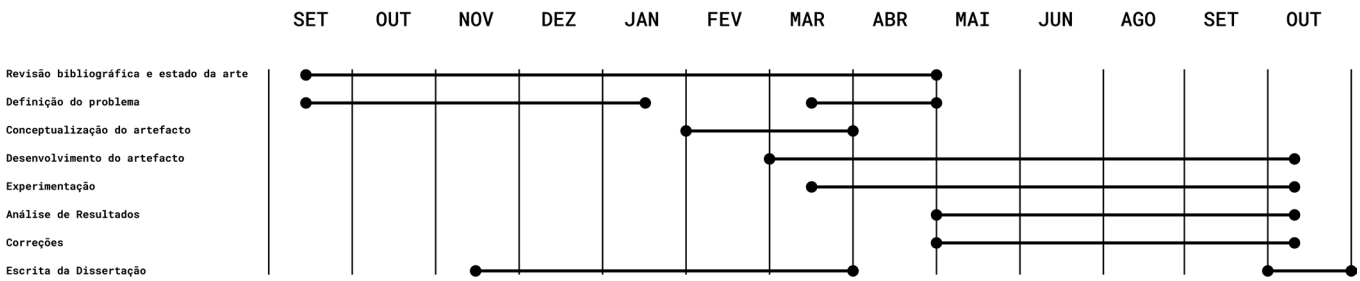
O trabalho começou em setembro com uma análise primária do texto "A Reinvenção da Leitura" de Ana Hatherly e com uma discussão inicial com vista a definir o foco principal do projeto. Nesta altura a Revisão Bibliográfica, na qual está integrada a pesquisa acerca do Estado da Arte, funcionou como suporte para a tarefa de Definição do Problema. Com base na pesquisa feita, em outubro chegou-se a uma conclusão inicial acerca do artefacto que seria produzido, produzir escrita assémica com tipografia. A partir desta ideia a Revisão Bibliográfica continuou.

A Escrita da Dissertação começou em novembro e prolongou-se inicialmente até fevereiro com principal foco na documentação do Estado da Arte.

A Conceptualização do Artefacto começou em fevereiro e previa-se que terminasse com o final do mês de março, durante esse período o principal foco seria o de pensar e procurar as características textuais que serão utilizadas para a manipulação dos textos, assim como a forma com o resultado final será apresentado. O Desenvolvimento do Artefacto começaria também durante este período de forma a poder colocar de imediato à prova a Conceptualização. Previa-se que o Desenvolvimento terminasse em abril.

O mês de maio estaria reservado para Análise de Resultados e Correções, e o mês de Junho seria de preferência utilizado apenas para a finalização de Escrita da Dissertação.

No entanto, por causa de implicações que não eram previsíveis no início do projeto, o trabalho prolongou-se até outubro de 2020. Desta forma, o diagrama a baixo representa com maior fidelidade o decorrer do projeto.



A conceptualização e o desenvolvimento começaram nas datas previstas, no entanto, houve o surgimento de inúmeros imprevistos, que motivaram o próprio desenvolvimento de ferramentas para que o fosse possível desenvolver o artefacto previsto. Assim sendo, o desenvolvimento, acabou por decorrer até ao mês de outubro, paralelamente com as tarefas de Experimentação, Análise de Resultados e Correções.

Estas circunstâncias, e a dimensão do trabalho inesperado que surgiu, obrigaram a que a Escrita da Dissertação tivesse menos tempo dedicado durante o decorrer do projeto. Tendo sido a última tarefa a ser levada a cabo.

Objetivos e métodos

O texto de Ana Hatherly, levanta mais questões, do que respostas. Propõe uma outra forma de olhar para o texto e para a letra, relembrando-nos de que estes são também imagens por si só.

Num campo que é descrito por um acumular de questões, e que se encontra definido entre duas disciplinas normalmente tomadas como absolutas, definir um objetivo que não seja também de questionar e explorar é praticamente impossível. Desta forma, parte do projeto foi também, descobrir o que seria o próprio projeto.

Um projeto de design, passa precisamente por isso. É um processo de pensamento, onde a experimentação e o erro definem o método de trabalho. O objetivo de um projeto de design é tornar algo mais claro e mais fiel a si mesmo.
Os textos-imagem de Ana Hatherly que acompanham o ensaio escrito têm também esse objetivo, tornar mais claro um problema.

A abordagem escolhida foi a de explorar ferramentas de design computacional para produzir artefactos gráficos que promovam outra forma de olhar para o texto, tal como os textos imagem de Hatherly e os restantes casos de estudo.

Inspirados pela escrita assémica definiu-se como objectivo o de criar generativamente e de forma automatizada tipografia que de alguma forma fosse entendida como escrita alfabética, mas que por outro lado fosse ilegível. Produzindo peças que comuniquem mais expressivamente que semanticamente. Através de uma entrada textual seriam criadas composições pseudo-textuais únicas.

Planeamento inicial da Abordagem

Para que o resultado seja entendido como legível, certas características do texto escrito têm de prevalecer. Nos vários exemplos apresentados nos casos de estudo existem sempre características que são herdadas do texto escrito que asseguram que o resultado final pode ser interpretado de alguma forma também como texto.

O desenho do tipo de letra com se escreve algo têm em si certa características que lhes conferem algo que pode ser entendido como um tom de voz, uma certa expressividade. Não existe uma fonte neutra que seja invisível aos olhos de quem a lê.

Procurando dar mais ênfase à expressividade visual de um texto, não faria sentido que a parte expressiva de um tipo de letra previamente utilizado fosse ignorada. Então de forma quer a preservar a expressividade inerente ao desenho das letras, e de forma a garantir que o nosso resultado final seria de alguma forma identificável como tipografia, foi decidido que o método a utilizar seria o de receber um *input* textual escrito numa qualquer fonte, e posteriormente, cada glifo desse *input* seria dissecado separando as suas partes anatómicas. Depois da separação, as peças resultantes seriam reconfiguradas, mantendo assim as características formais da fonte utilizada no *input* inicial, e de alguma forma assegurando que o resultado poderia ser interpretado como algo textual e não apenas uma expressão gráfica.

A abordagem prevista inicial pode ser descrita da seguinte forma:

1. Input
    a. Texto é introduzido no sistema
2. Dissecação dos Glifos.
    a. Localização de partes anatómicas
    b. Corte
3. Reconfiguração
    a. Conexão de partes anatómicas em novos glifos

1. INPUT

a. Texto é introduzido no sistema

O objetivo do projeto é desenvolver uma ferramenta dinâmica que não produza apenas um, mas muitos resultados diferentes. Para isso, decidiu-

se que o sistema deveria ser capaz de receber qualquer tipo de texto, escrito com qualquer tipo de letra. Isto seria escolhido por um utilizador. Para diferente introdução de texto no sistema, um resultado diferente deve ser retornado.

2. DISSECAÇÃO DOS GLIFOS

O desenho de um tipo de letra, é um exercício de coerência e contraste. Na criação de um alfabeto tipográfico, os diferentes glifos, têm de ser semelhantes o suficiente para que sejam reconhecidos como um sistema e não objetos individuais, e têm de ser diferentes o suficiente para que seja possível identificar cada glifo com o caracter que representa e para que seja possível distinguir entre glifos da mesma fonte.

Assim sendo, é natural que existam partes dos seus desenhos que sejam partilhadas entre diferentes glifos. Desta maneira, de forma a tornar possível uma comunicação detalhada sobre a anatomia de um glifo, as suas partes anatómicas podem ser identificadas e nomeadas. Querendo manter uma aparência gráfica semelhante à de uma fonte, faz sentido basearmo-nos na identificação das partes anatómicas dos seus glifos para o seu desenho.

a. Localização de partes anatómicas

Depois do texto ser introduzido será necessário localizar e identificar as partes anatómicas de cada glifo. Tendo em conta que lidamos com qualquer tipo de glifo, de qualquer fonte, é preciso desenvolver um método que funcione de forma generalizada.

i. Deteção do esqueleto
Para quase qualquer glifo é possível imaginar uma linha que passa pelo interior dos seus traços e que define a sua estrutura, como um esqueleto. Para simplificar os glifos e desenvolver um método abrangente, o esqueleto de cada glifo deverá ser calculado.

ii. Identificação das "articulações"
Depois de detetar o esqueleto é preciso identificar zonas onde 3 ou mais linhas se intersecção, isso representará uma "articulação". O esqueleto será então quebrado nestas articulações resultando o que serão os "ossos" de cada uma das partes anatómicas do glifo.

b. Corte do glifo

Utilizando o conjunto de "ossos" resultantes, deverá ser possível fazer cortes nas zonas das "articulações" e assim separar as partes anatómicas do glifo.

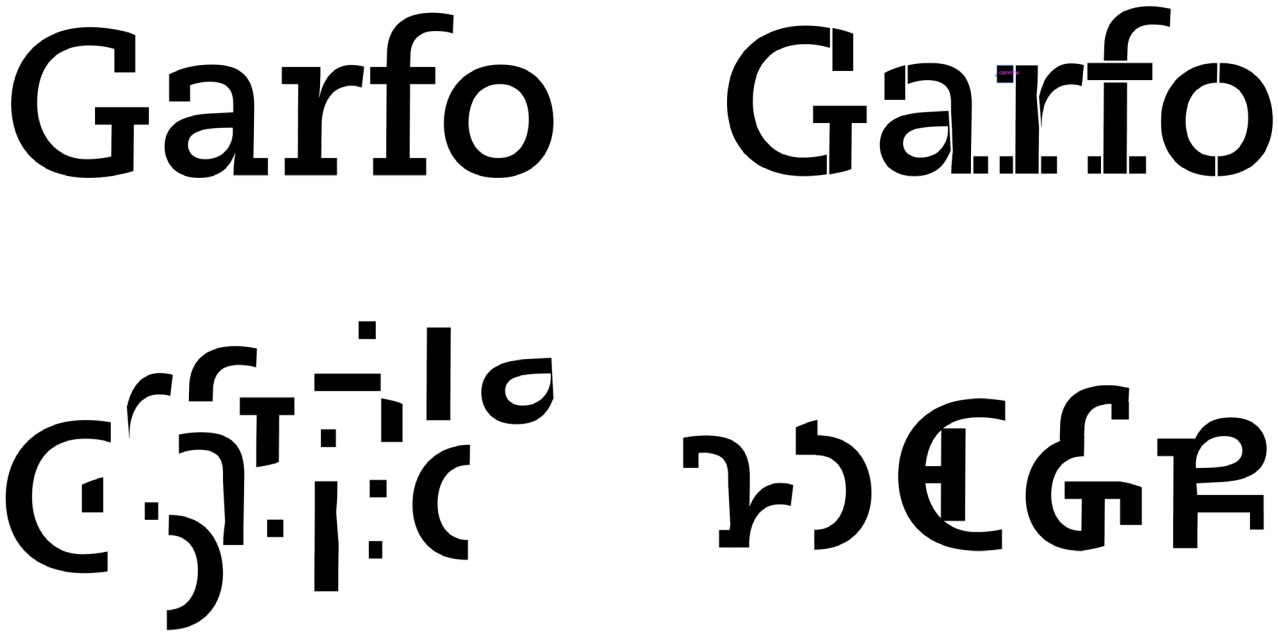
Tendo acesso às partes anatómicas isoladas, identificar-se-ão as zonas de conexão. E depois, aleatoriamente, ou de acordo com alguma parametrização, as partes anatómicas serão conectadas com outras provenientes de outros glifos.

Existirá ainda a necessidade de filtrar as conexões entre peças de forma a manter características tipográficas aceitáveis, como por exemplo: altura dos descendentes, altura dos ascendentes, altura-x, posição do glifo em relação à linha de base, número máximo e mínimo de partes por glifo, tracking, leading, etc. Isto será importante para garantir que o resultado não seja, por exemplo, um único mega-glifo composto por todas as peças anatómicas provenientes do input. No entanto, esta filtragem pode e deverá ser parametrizada e liberada experimentalmente de forma a procurar resultados alternativos.

Resumidamente, a ferramenta faz as perguntas:

Neste tipo de letra, aplicando as suas regras de desenho, que outros mais glifos poderiam existir para além dos que representam as letras do alfabeto?

De que outra forma se pode ler um texto?



img 23 (+ próxima pág.)
Esquematização do funcionamento da ferramenta a desenvolver.

Desenvolvimento

Inicialmente, várias partes do processo de implementação decorreram em paralelo entre si e enquanto investigação técnica e conceptual também decorriam. Cada parte da abordagem foi desenvolvida separadamente sem nenhuma ordem em particular. À medida que problemas e imprevisibilidades surgiram durante o decorrer do processo o método tornou-se mais sistemático.

O número de imprevistos e as limitações das ferramentas disponíveis que se manifestaram ao longo do percurso impossibilitaram a finalização total da ferramenta idealizada. No entanto, essas mesmas limitações criaram a oportunidade para a sua resolução e para a construção de ferramentas que não só tornam o resultado final deste projeto mais próximo, como também poderão ser úteis noutros projetos da área.

IMPLEMENTAÇÃO DA DETEÇÃO DE ESQUELETOS

Implementação do Algoritmo K3M

O artigo "*K3M: A universal algorithm for image skeletonization and a review of thinning techniques*" apresenta um algoritmo de deteção de esqueletos de formas geométricas representadas em *bit-map*. Pelo que, para ser utilizado para detetar os esqueletos de formas vectoriais, estas têm de ser primeiramente convertidas em imagens *bit-map*.

O algoritmo funciona por eliminação iterativa dos *pixels* fronteiriços (*pixels* pertencentes à forma que tenham na sua vizinhança pelo menos um *pixel* vazio) de uma forma representada numa matriz (imagem *bitmap*), até apenas sobrar uma forma cuja espessura é de apenas 1 *pixel*, a qual já não é afectada pelo processo do algoritmo. (Saeed et al., 2010)

Este algoritmo foi selecionado por estar referido no artigo científico "*Character Recognition Based on Skeleton Analysis*" (Sarnacki et al. 2018) onde é utilizado na deteção de formas de letras, uma tarefa semelhante à que nos propúnhamos, e onde aparentemente obtém bons resultados. Por isto foi o algoritmo escolhido inicialmente.

O algoritmo K3M foi implementado em Processing de acordo com as instruções fornecidas no artigo em conjunto com um programa de teste que pode ser descrito da seguinte forma:

1. Receber um input textual
2. Representar o input com um tipo de letra escolhido
3. Converter o desenho vectorial dos glifos em formas bitmap
4. Aplicar o algoritmo
5. Desenhar o resultado do algoritmo sobreposto à forma inicial para análise

Os resultados apresentados foram bons, mas não completamente satisfatórios.

- Junto de alguns vértices das formas introduzidas no algoritmo, artefactos adicionais eram criados, o que poderia sugerir a existência de partes anatómicas onde estas não existiam (serifas por exemplo).

- Em formas simples, como por exemplo nas pintas das letras i e j o esqueleto (que neste caso deveria ser apenas um ponto ou uma pequena linha) era deslocado para o canto inferior esquerdo.

img 24 (esquerda)
Deteção de esqueleto utilizando o algoritmo K3M a grande escala

img 25 (direita)
Deteção de esqueleto utilizando o algoritmo K3M a pequena escala

Implementação utilizando biblioteca externa em python

Para comparação, recorreu-se a uma biblioteca externa open-source para processamento que imagem escrita em python, "*scikit-image*". Esta biblioteca, entre outras funções, oferece diferentes funções, previamente implementadas, para deteção de esqueletos baseadas em diferentes algoritmos alternativos ao algoritmo utilizado inicialmente, *K3M*. Os resultados gráficos apresentados por este método mostraram-se mais consistentes em comparação com o resultado do algoritmo *K3M*.

img 26
Deteção de esqueleto utilizando a biblioteca *scikit-image*

No entanto, o processo de exportação e importação das imagens para serem utilizadas por esta biblioteca externamente e depois mais tarde recebidas mostrou-se muito mais lento que a implementação direta do algoritmo.

Como a deteção dos esqueletos dos glifos só teria de ser feita uma única fez e depois poderia ser reutilizada sempre que esse mesmo glifo fosse selecionado pelo utilizador este atraso de resposta por parte do algoritmo não nos pareceu impeditivo do bom funcionamento da nossa ferramenta. Por isto, o problema de deteção de esqueletos foi marcado como resolvido e o foco do desenvolvimento passou para outros requisitos do projeto

É de notar que mais à frente será apresentado outra abordagem para deteção de esqueletos de formas que acabou por ser a utilizada. No entanto, de forma a manter a coerência cronológica neste capítulo optou-se por escrever sobre as vantagens e problemas desta outra abordagem noutra secção.

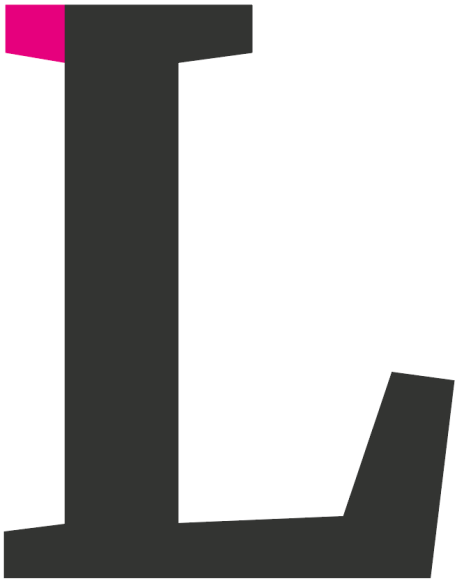
IMPLEMENTAÇÃO DA RECONFIGURAÇÃO

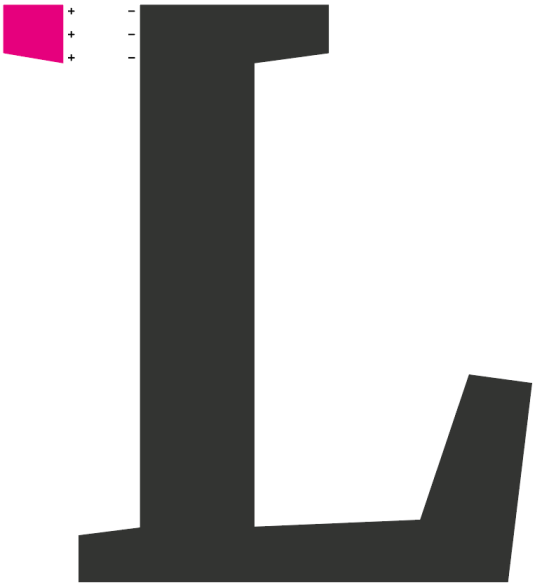
O processo de reconfiguração é a parte do projeto que é responsável por receber diferentes partes anatómicas de glifos e conectá-las de uma forma tipograficamente lógica.

O conceito mais importante deste processo é o de zona de conexão, que foi tratada metaforicamente como uma zona magnética. Ou seja, a ideia de que a conexão entre duas partes anatómicas não pode acontecer aleatoriamente, mas sim entre duas zonas específicas de cada peça que se atraem mutuamente.

Por exemplo, uma serifa terá apenas uma zona magnética pela qual poderá ser conectada. No entanto outras partes anatómicas, como uma haste poderão ter várias zonas magnéticas onde esta seria se poderá conectar.

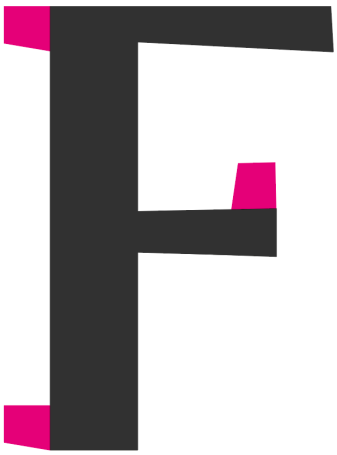
A peça poderá ser rodada para se conectar corretamente. O mais importante é que as conexões apenas aconteçam entre zonas magnéticas.

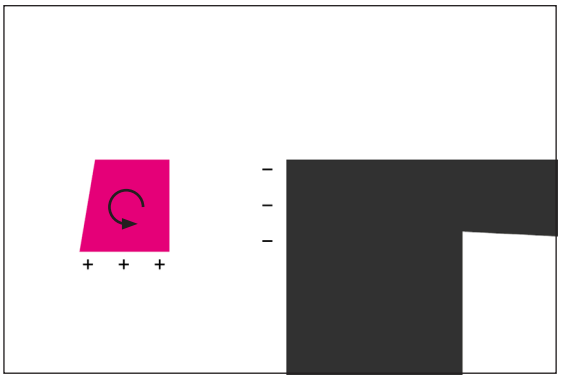




img 27 (esquerda)
Identificação de uma parte anatómica, uma serifa neste caso.

img 28 (direita)
A separação da serifa da haste revela o magnetismo que atrai a serifa para aquela posição.





img 29 (topo esquerda)
Possiveis zonas de ligação da serifa com a haste de um *F*, e campos magnéticos.

img 30 (topo direita)
Possiveis ligações da serifa com uma letra *F*.

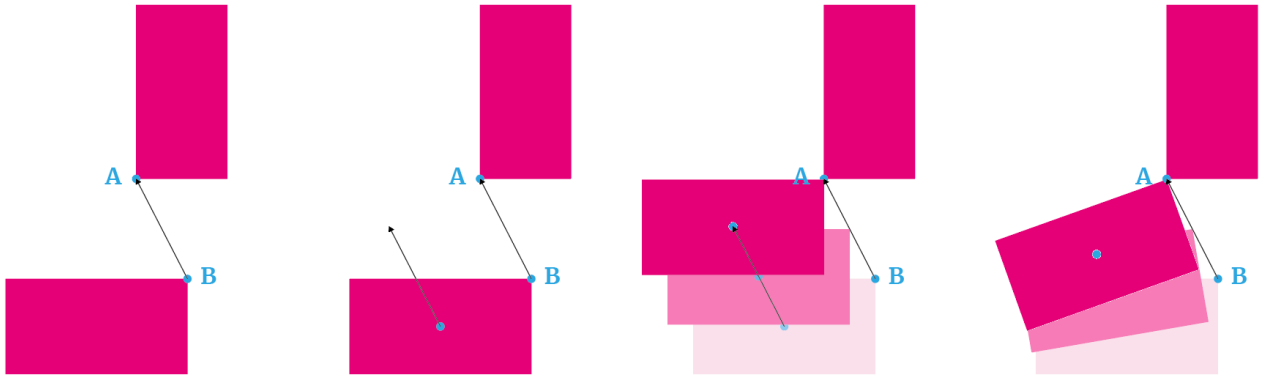
img 32 (baixo)
Devido à atração magnética uma serifa que esteja desorientada em relação à zona de ligação, roda naturalmente para se alinhar.

Desta maneira, a abordagem para conexão de peças anatómicas escolhida foi a de simulação física, simulando precisamente a atração magnética entre diferentes partes anatómicas.

As zonas magnéticas seriam identificadas, as partes anatómicas baralhadas, e posteriormente as forças magnéticas calculadas e simuladas, resultando em principio numa nova reconfiguração tipograficamente lógica.

Implementação

A simulação da atração magnética têm se ser feita calculando as forças exercidas por pontos magnéticos de uma peça num ponto magnético de uma outra peça. E com esta informação o efeito provocado por esta força na peça tem de ser calculado, normalmente representado não só pela variação da posição da peça, mas também a variação da rotação da peça.



img 32 (esquerda)
Atração entre o ponto A e B.

img 33 (centro esquerda)
Vetor de delocamento simples aplicado ao centro de massa da peça.

img 34 (entro direita)
Resultado do delocamento simples aplicado ao centro de massa da peça.

img 35 (direita)
O pretendido, deslocamento + rotação.

Como se pode ver no exemplo, existindo uma atração feita pelo ponto A ao ponto B o expectável não é apenas o deslocamento da peça de baixo (img 34), mas também a rotação da peça de baixo (img 35). Isto acontece porque a força está a ser exercida num ponto específico da peça, e não no seu centro de massa.

A implementação deste fenómeno físico é essencial para a implementação correta de um sistema de simulação magnética que permita que naturalmente as peças rodem e se alinhem corretamente com as zonas para as quais estão a ser atraídas.

Outro conceito importante é o de deteção de colisões.

img 36 (esquerda)
Identificação do ponto A e B.

img 37 (centro esquerda)
Atração entre o ponto A e B.

img 38 (entro direita)
Resultado do deslocamento sem deteção de colisões.

img 39 (direita)
Resultado do deslocamento com deteção de colisões.

A colisão entre duas peças tem de ser calculada de forma a evitar que a atração entre dois pontos de peças diferentes seja feita sempre pelo trajeto mais curto e produza sobreposições entre as peças (img 38). Se as peças se tocarem, a conexão deve ser evitada (img 39).

Para implementação deste tipo de comportamento foi utilizada uma biblioteca de simulação física. Neste caso, a biblioteca escolhida foi a implementação para Processing da biblioteca box2D.

O box2D é uma biblioteca de simulação de física que existe em várias linguagens de programação, é amplamente utilizada em diversos tipos de projetos, principalmente em jogos.

Esta biblioteca oferece a possibilidade de associar propriedades físicas a objetos e definir as condições em que estes se encontram.

Ao escolher esta biblioteca para o nosso projeto, não contámos com limitações que não eram previsíveis. Uma dessas limitações pode ser descrita da seguinte forma: Na maioria dos casos em que esta biblioteca é utilizada, a precisão não necessita de ser exacta. Por isso, para otimizar o desempenho da biblioteca, cada forma apenas pode ter um máximo de 8 vértices, isto torna o calculo de intersecções entre objetos muito mais simples ao evitar calcular intersecções entre curvas. Este polígono é “conectado” com a forma inicial e é através dele que as colisões são calculadas. No entanto no nosso caso a precisão de uma colisão é algo importante, uma vez que a junção de peças curvas seria algo bastante comum. Pode ver-se na img 40 dois exemplos do possível resultado da aplicação direta da biblioteca.

img 40 (esquerda)
Métodos de simplificar formas circulares e suas consequências.

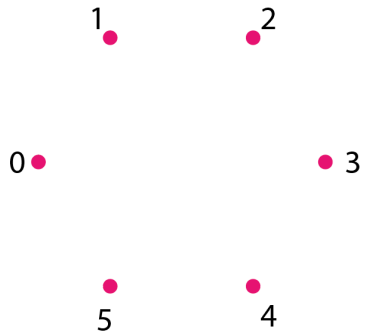
Considerando por exemplo uma letra R que esteja separada em três partes. Existem duas possibilidades para a aproximação da forma de cada uma dessas partes em polígonos com um máximo de 8 vértices. A primeira é a de distribuir os vértices do polígono ao longo do contorno da forma inicial (exemplo de cima). Isto resulta numa polígono que não engloba na totalidade a forma inicial. Assim sendo, como é através destes polígonos que as colisões são calculadas, uma colisão resultaria em sobreposições entre as formas iniciais.

Outra possibilidade para criar um polígono aproximado de uma forma é colocando os vértices fora da forma, no prolongamento de tangentes às curvas, de maneira que a forma inicial fique toda englobada dentro do polígono (exemplo da linha de baixo). No entanto, esta abordagem não permite que numa colisão as formas iniciais se cheguem a tocar uma vez que os polígonos envolventes colidem sempre antes das formas iniciais se tocarem.

Antes de abandonar esta metodologia de reconfiguração de partes anatómicas tentou-se arranjar alternativas para a maneira como as intersecções são calculadas de forma a que fosse possível continuar a utilizar a biblioteca.

A primeira tentativa de solução passou por estruturar um algoritmo que dividisse as formas que tenham mais do que 8 vértices em sub-peças que seriam tratadas em grupo. No entanto as figuras têm de ser cortadas de acordo com uma metodologia de forma a assegurar que a forma da peça se mantêm e que o número de vértices das figuras resultantes é diminuído consideravelmente.

Expondo o problema graficamente:

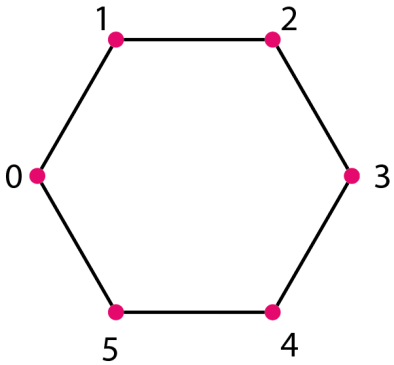


img 41
Pontos pertencentes a um forma.

Uma forma é composta por uma lista ordenada de coordenadas

0 - (x0, y0), 1 - (x1, y1), 2 - (x2, y2), 3 - (x3, y3), 4 - (x4, y5), 5 - (x5, y5)

Essas coordenadas são conectados por linhas ou curvas (neste caso, e devido à natureza do box2d apenas linhas, uma vez que as colisões têm de acontecer entre polígonos) pela ordem do seu identificador na lista.



img 42
Ligação de pontos pertencentes a um forma.

Embora esta forma não tivesse a necessidade de ser dividida por ter menos de 8 vertices é útil para uma exemplificação mais simples.

Dividir uma forma - sem criar mais pontos - passa por selecionar dois

items da lista e criar duas novas sub-listas começando e terminando nesses items: Da seguinte forma:

Escolhendo (1) e (3):

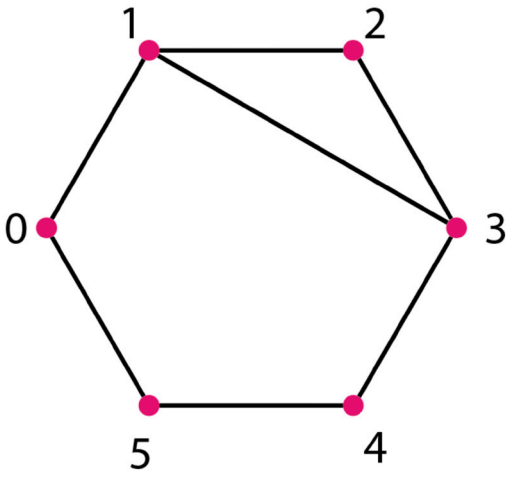
A - 1 - (x1, y1), 2 - (x2, y2), 3 - (x3, y3)

B - 0 - (x0, y0), 1 - (x1, y1), 3 - (x3, y3), 4 - (x4, y4), 5 - (x5, y5)

as listas resultantes são:

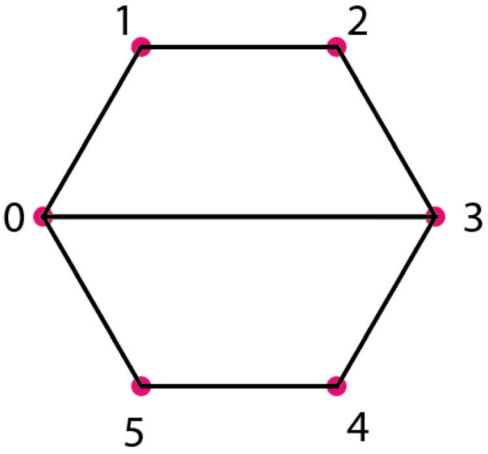
A - 0 - (x1, y1), 1 - (x2, y2), 2 - (x3, y3)

B - 0 - (x0, y0), 1 - (x1, y1), 2- (x3, y3), 3 - (x4, y4), 4 - (x5, y5)



img 43
Abordagem para separação

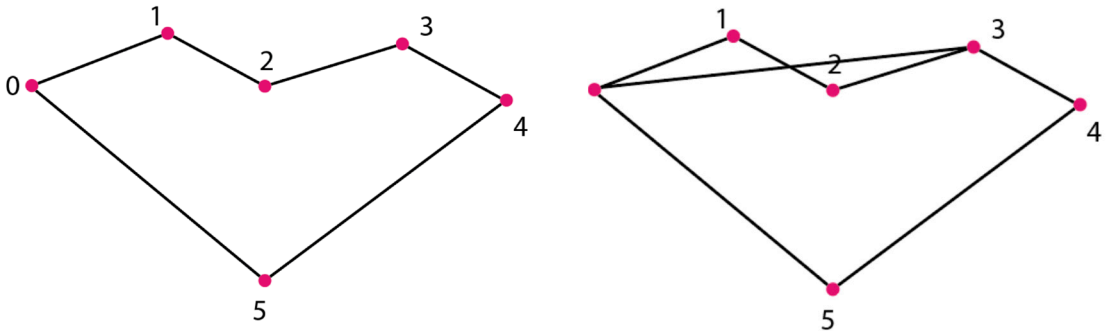
Assim sendo, esta forma dá a indicação de que para a dividir em partes semelhantes de forma a reduzir da maior forma possível o número de pontos de cada parte, o que se poderia fazer seria cortar a forma no ponto (0), e no ponto central da lista, neste caso (3).
Da seguinte forma:



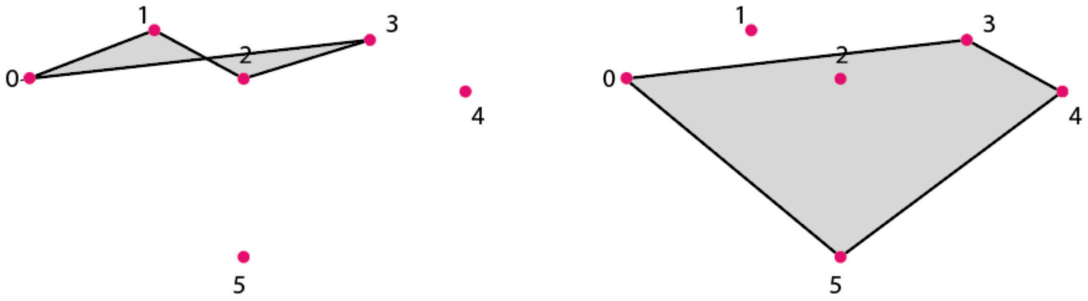
img 44
Abordagem para separação

Resultando em duas formas com quatro pontos cada.

No entanto, este algoritmo não é aplicável a qualquer forma:



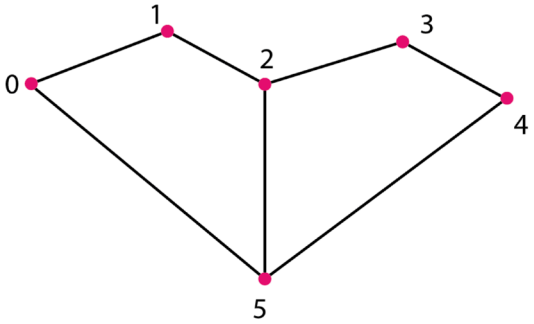
img 45
Corte de formas não puramente convexas



img 46
Resultado do corte de formas não puramente convexas

No exemplo a cima as formas resultantes são as seguintes, que somadas não representam a forma inicial

Neste caso, para a forma dada, o corte desejável seria algo semelhante a isto:

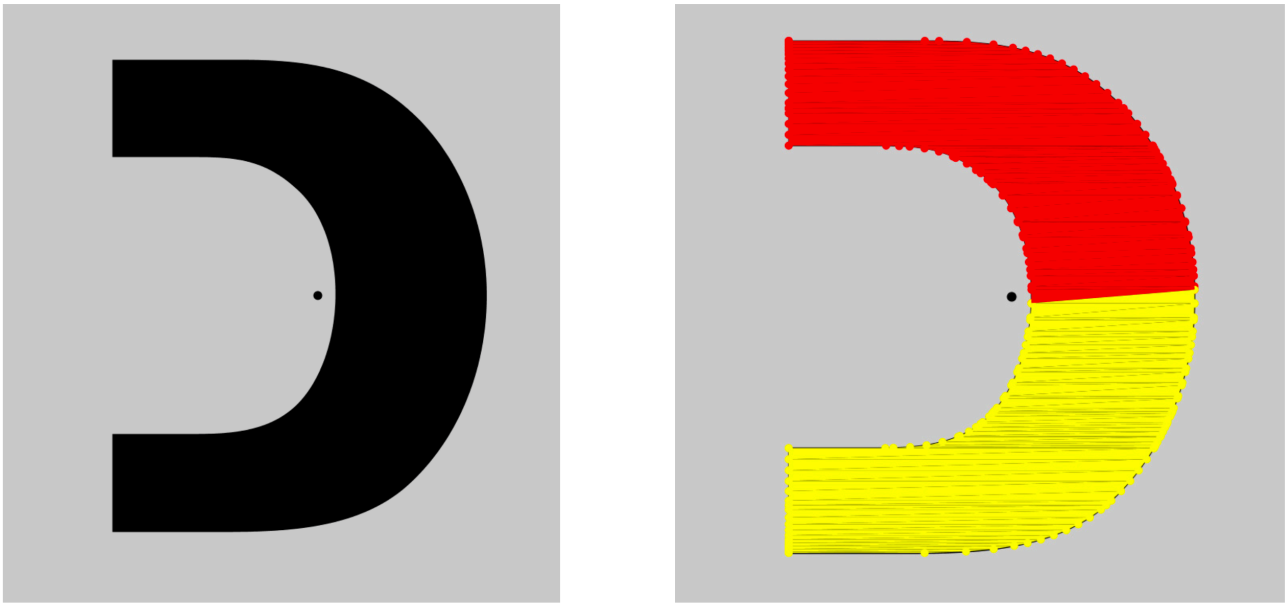


img 47
Intensão de resultado de corte de formas não puramente convexas

Para obter este resultado optou-se por utilizar primeiro reorganizar a forma numa mesh de triângulos:

Nesta rede triangular é então possível utilizar a sua nova organização para selecionar os pontos de corte que seriam sempre dois pontos consecutivos da zona mais central da lista, neste caso (2 e 3), (que correspondem aos pontos 2 e 5 na forma original).

Implementou-se então este algoritmo que faz a seguinte divisão:



img 48
Corte de forma com base em rede de triângulos.

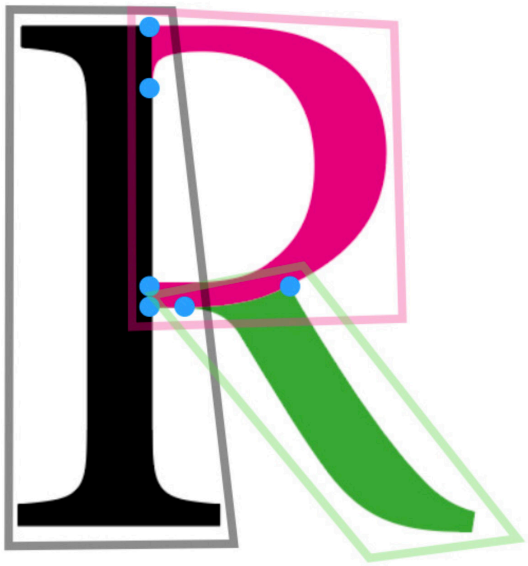
No entanto cada uma destas forma continua a ter mais do que 8 vértices, e teria de ser dividida continuamente até nenhuma das suas partes ter um número de vértices inferior a 8.

Esta abordagem continua a não ser fiável, por isso a técnica teve de ser modificada.

Solução 2

Não obtendo bons resultados com os métodos anteriores, optou-se por abdicar do calculo de colisões, e utilizar apenas os pontos de junção para obter um bom resultado.

Se os pontos onde as partes se juntam forem guardados, pode-se simplesmente criar polígonos envolventes das formas e ignorar a necessidade de deteção da colisão.

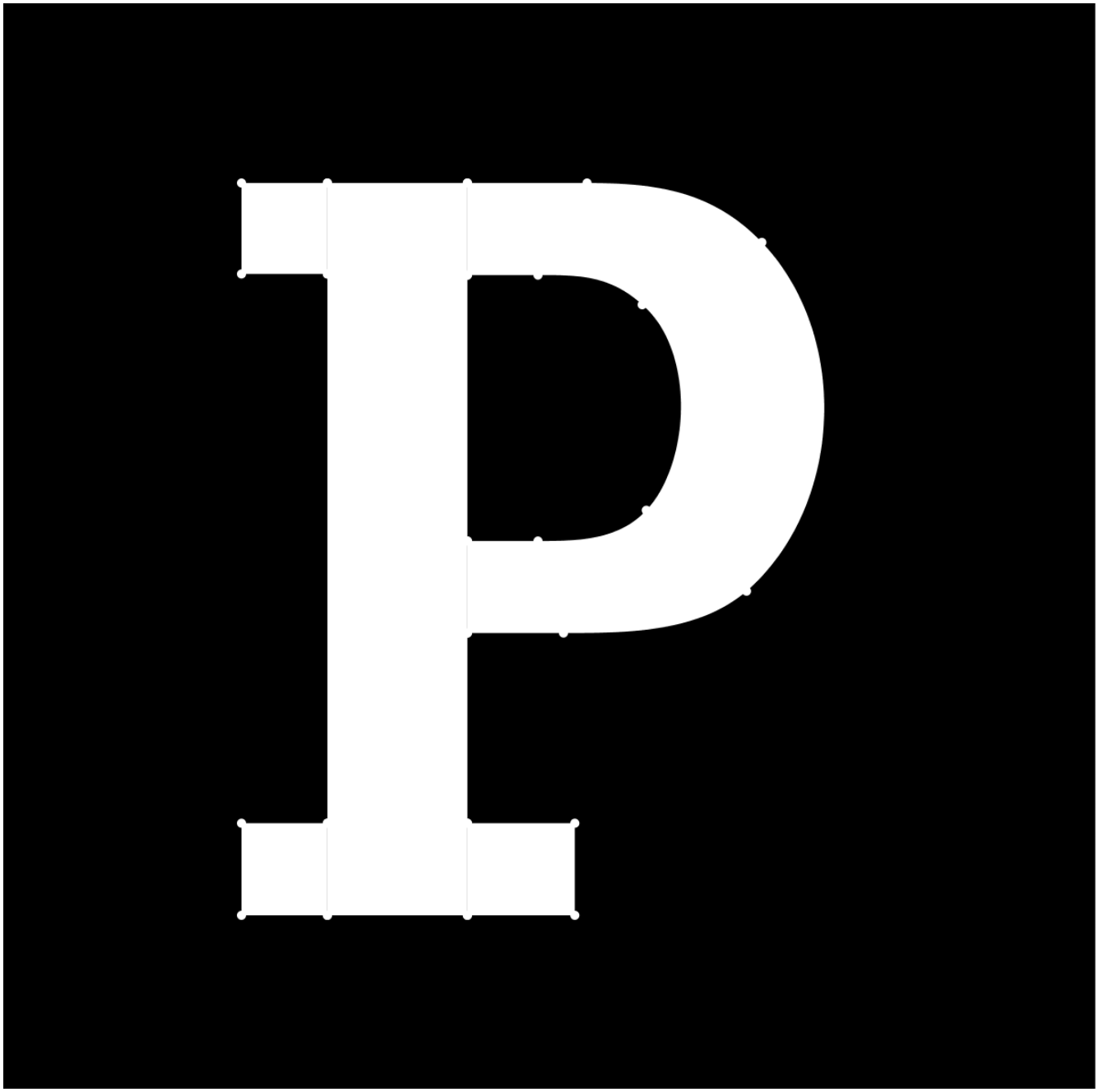


img 49
Glifo com partes anatómicas e seus objetos envolventes, e pontos de conexão identificados.

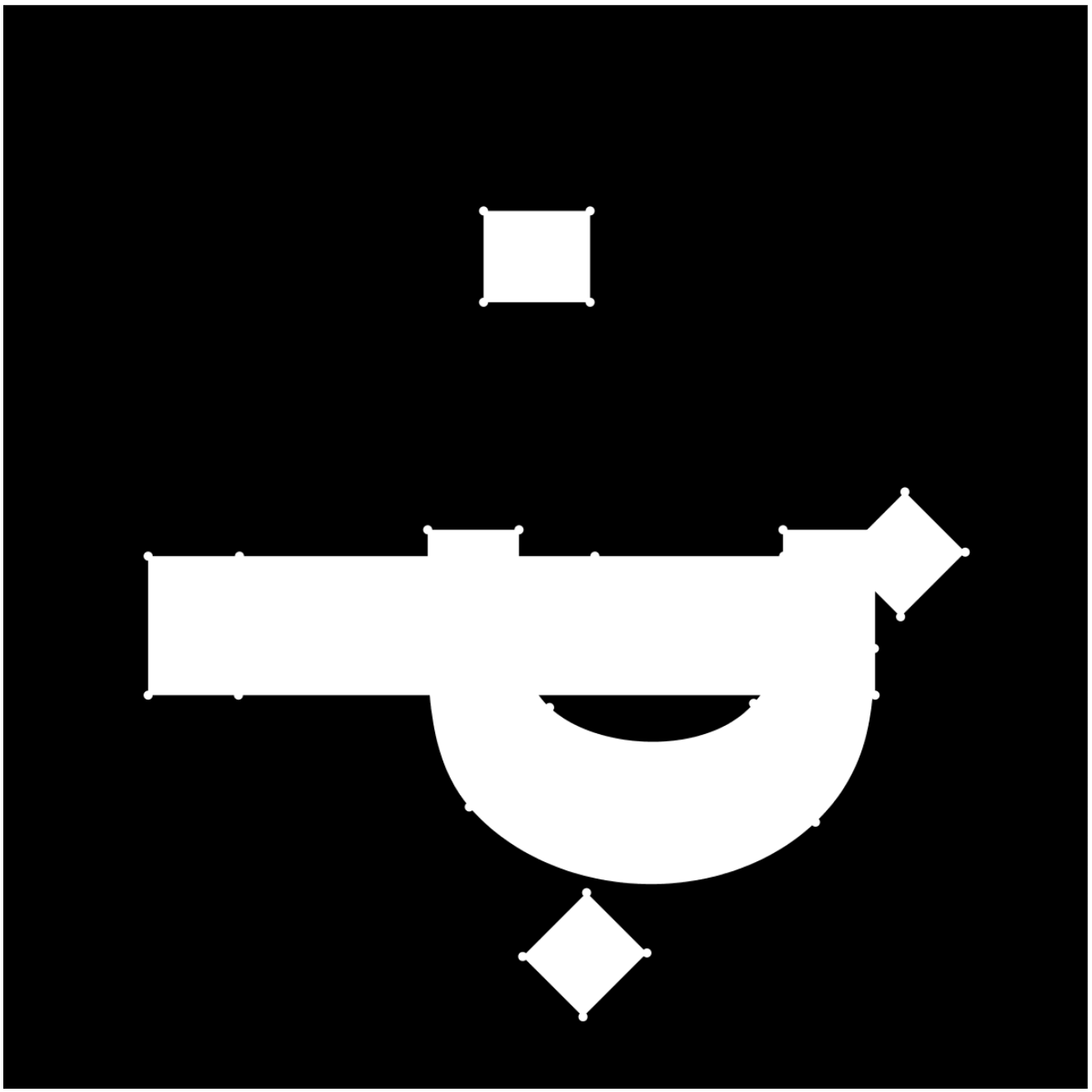
Desta forma os objetos de colisão podem ser sempre quadriláteros, que se podem sobrepor. Assim, não existe a necessidade de partir cada peça em partes com menos vértices. Esta abordagem pode funcionar desde que a ligação seja feita sempre por junção dos pontos de conexão (neste exemplo, a azul).

Embora aparentemente promissora, esta abordagem não produziu resultados com boa qualidade:

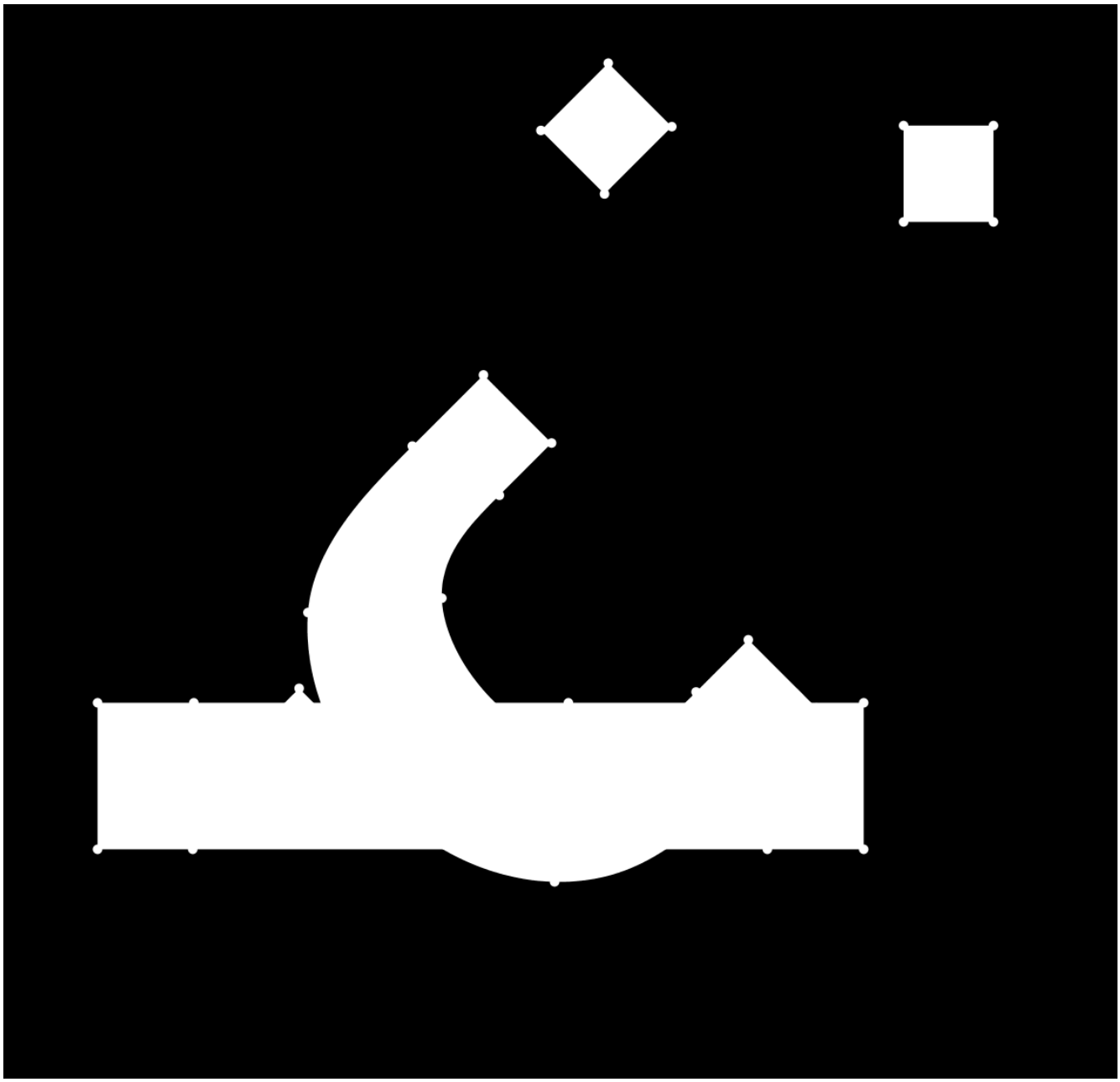
Primeiro:
As partes anatómicas selecionadas do “P” são reconfiguradas para os resultados seguintes, que demonstram algum interesse do ponto de vista gráfico, por se assemelharem a glifos de caracteres árabes:



img 50
Letra P previamente fragmentada



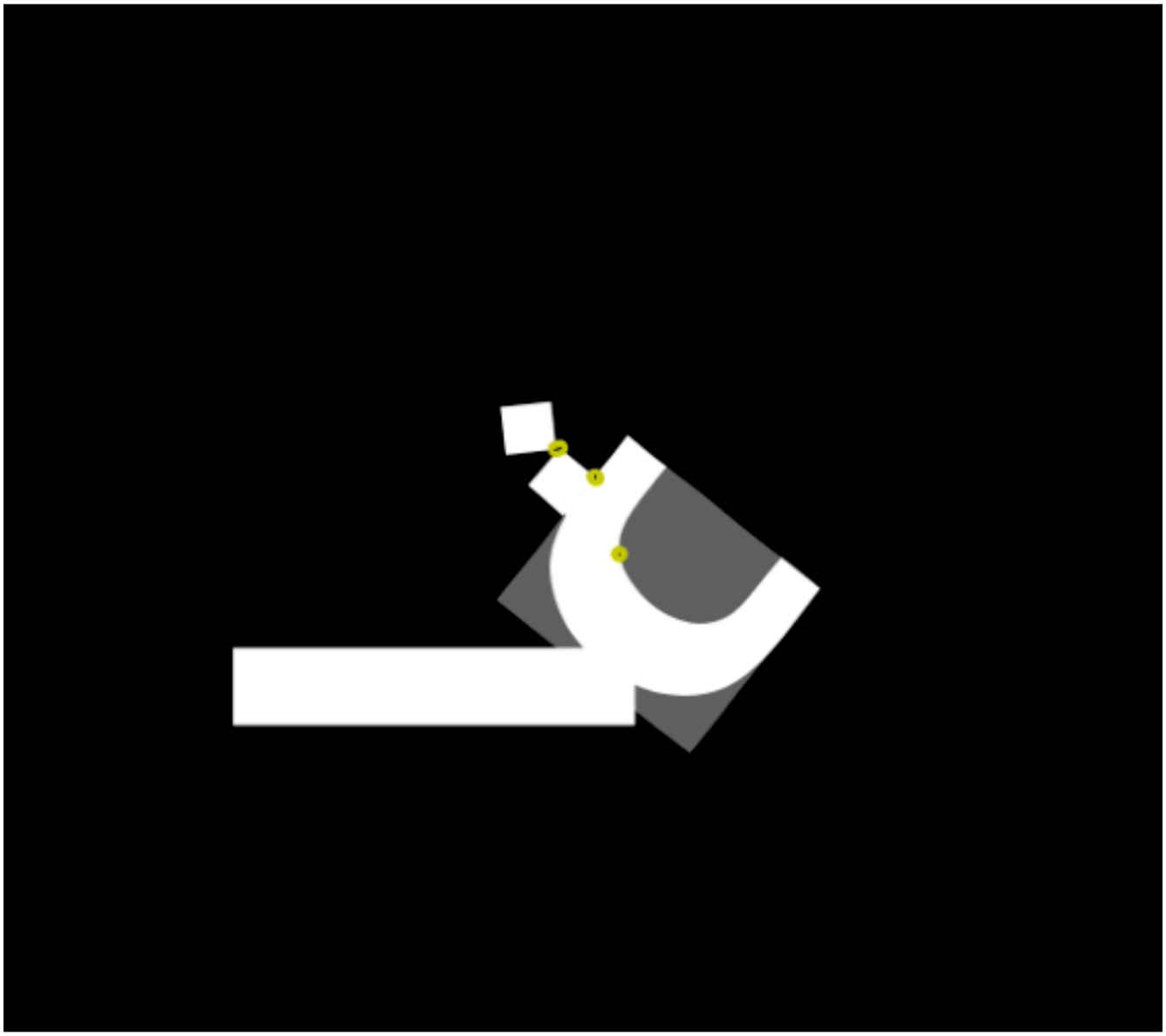
img 51
Reconfiguração aleatória



img 52
Reconfiguração aleatória

img 53
Reconfiguração aleatória

Depois, utilizando a física implementada do box2D, pontos de conexão de cada parte anatómica atraem outro ponto anatómico de uma outra parte:



img 54
Reconfiguração com base em atração

Pode ver-se que, mesmo fazendo a conexão pelos pontos predefinidos existem sobreposições de mais para o resultado ser interpretado como um caracter tipográfico. A ligação entre as partes anatómicas está a ser feita pelos pontos corretos, no entanto como não existe deteção de colisões, não existe forma de evitar um ponto seja atraído por outro que está do lado oposto da parte anatómica. Resultando assim na sobreposição das formas.

Como já foi referido no início, a biblioteca *box2D* é uma biblioteca dedicada à simulação de física (é utilizada por exemplo no jogo Angry Birds), e por isso, aplica conceitos e tem particularidades específicas desse meio. Algumas dessas, particularidades que previmos e desejávamos para a concretização do trabalho (como por exemplo a aplicação de forças em partes específicas de objetos, para as conexões entre partes anatómicas), mas também outras que não previámos e que vieram a impossibilitar a sua utilização no projeto. Um desse conceitos presente na box2D é o conceito de gravidade.

Num ambiente de simulação física, como um video-jogo, a gravidade é um conceito que está quase sempre presente, por exemplo para que um personagem volte a cair quando salta. A biblioteca box2D oferece a possibilidade de configurar os valores dessa força gravitacional para simular diferentes ambientes, com gravidade mais forte ou mais fraca, para baixo ou para cima, etc.

No nosso caso, o conceito de gravidade não seria muito facilmente aplicável uma vez que as letras normalmente não caem das folhas de papel ou dos ecrãs. No entanto, depois da implementação das conexões entre partes anatómicas, percebeu-se que no meio das várias formas como se pode manipular os valores da gravidade para o ambiente físico do box2D, não é possível anular completamente a gravidade. Pelo menos sem comprometer o funcionamento da restante física. Ao colocar os valores da gravidade a 0.0, as letras já não caiam, mas também não se juntavam nas novas posições. Ao anular a gravidade, as partes anatómicas da letra não eram reconectadas. Não nos foi possível diagnosticar qual seria o problema no programa implementado pelo que só é possível concluir que é um problema que sai fora do âmbito da *box2D* e que a os valores da gravidade têm algum papel na restante biblioteca.

Como para o nosso projeto não é possível que as formas estejam em constante movimento, e como é necessários que as conexões aconteçam, a biblioteca *box2D* e todo o trabalho feito baseado no conceito de atração magnética/física entre partes anatómicas passou a ser inútil. E por isto a utilização da biblioteca *box2D* foi abandonada.

ESTUDO PRÁTICO DA ANATOMIA DA LETRA

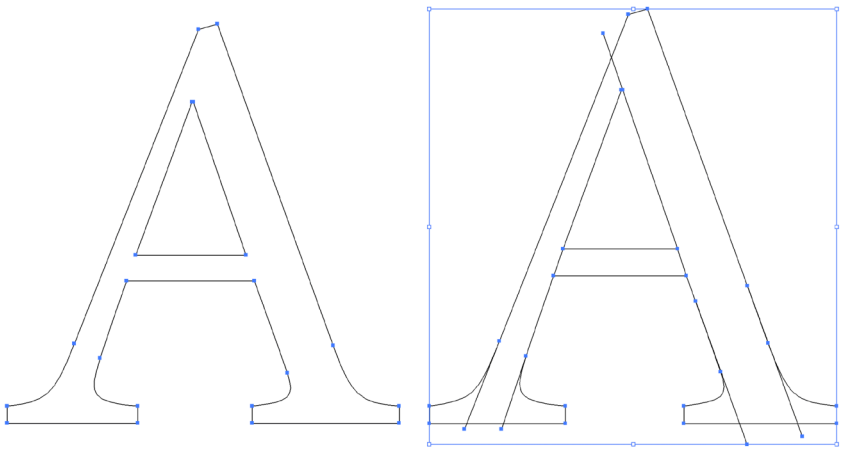
Os resultados do processo de desenvolvimento com base num modelo físico, embora tenham fracassado, provocaram o surgimento de novas questões acerca da metodologia de corte e reagrupamento de partes anatómicas. A estrutura e hierarquia das partes anatómicas dentro de um glifo não são aleatórias, uma haste não se junta com outra haste simplesmente, uma serifa não existe sem uma outra parte anatómica, e ainda há partes anatómicas que se subdivididas se assemelham a outras partes anatómicas.

Para uma análise mais aprofundada da anatomia do tipo de letra procedeu-se a um exercício manual de corte dos glifos maiúsculos e minúsculos do alfabeto latino de uma fonte.

img 55
Maiúsculas do alfabeto latino dissecadas

img 56
Minúsculas do alfabeto latino dissecadas

A dissecação dos glifos foi feita com base nos pontos definidores do desenho das formas o mais possível tentando proceder de uma forma programaticamente lógica com o objetivo de tentar compreender os problemas que o algoritmo de corte teria de ultrapassar, quais as escolhas que teria fazer e que padrões poderiam ser encontrados.



img 57
Exemplo da metodologia de corte utilizada

Os pontos de desenho da letra são importantes porque ajudam a tomar decisões acerca de onde fazer os cortes. Com base na posição dos pontos linhas são prolongadas para cortar o glifo de uma forma que possa ser descrita programaticamente.

Durante este exercício foi utilizado pela primeira vez uma metodologia de agrupamento de partes anatómicas baseada na hierarquização das partes da letra, tratando a forma como um todo que poderá ser quebrado a vários níveis.

img 58
Diferentes níveis de quebra do glifo

Por exemplo, uma letra A, poderá ser quebrada inicialmente, no que poderá ser considerado 3 partes, e num segundo nível algumas dessas partes poderão ser elas quebradas, havendo assim uma noção de peça prevalecente sobre outra e de hierarquia anatómica.

Este método análise anatómica permite um mais fácil relacionamento de fontes serifadas com fontes não serifadas. Desta forma, ambos os A's são representados num primeiro nível de corte por 3 partes, que num dos glifos podem ser repartidos mais uma vez. Isto parece-me ser uma definição mais lógica do que assumir simplesmente que um A de um tipo de letra serifado tem mais 4 (ou 2, se se considerar a serifa como bilateral (Amado, 2011) partes anatómicas que um não serifado.

Outro problema levantado por este exercício foi o da multiplicidade de formas pelas quais um glifo pode ser dividido.

img 59
A serifado e *A* não serifado com as respetivas partes anatómicas

Por exemplo, o glifo utilizado como referência a cima poderia também ser cortado da seguinte forma:

img 61
A serifado com haste esquerda completa

Trocando o corte que separa as duas hastes no ápice do A, ficando a haste azul completa desta vez.

Ou ainda:

img 60
A serifado com serifas bilaterais

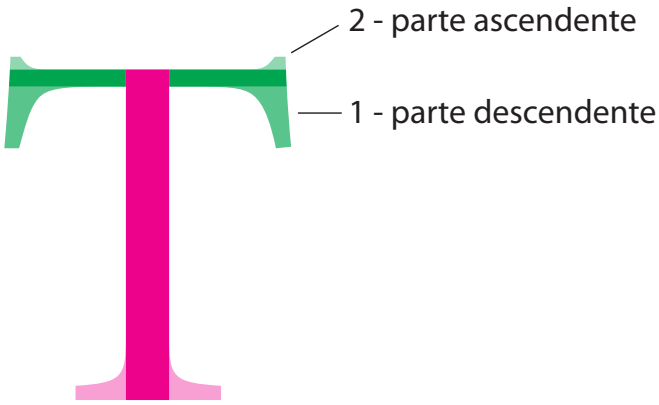
Separando as serifas com cortes horizontais e tratando-as como serifas bilaterais (que poderiam ou não ser uma fez mais repartidas aplicando a ideia de hierarquia)

Qualquer uma destas abordagens de corte tem as suas vantagens e desvantagens, por exemplo nesta última abordagem, a separação das serifas da haste, torna a haste mais curta, fazendo com que uma peça de hierarquia superior não possa ser utilizada independentemente das serifas. No entanto oferece a possibilidade de tratar a serifa como um todo.

Por isto um estudo e classificação das metodologias de corte possíveis foi feito à volta do desenho da letra T, que permite demonstrar muitas destas problemáticas.

METODOLOGIAS DE CORTE DE UM GLIFO

ABORDAGENS DE JUSTAPOSIÇÃO



img 62
Corte longitudinal da serifas

Vantagens desta abordagem:

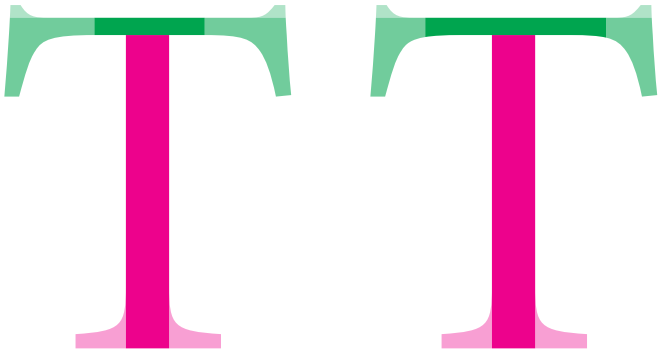
· Cada lado do travessão fica completo, e faz sentido sem a serifa

· A haste fica completa, logo também ela faz sentido sem as serifas

Desvantagens:

· O travessão fica separado em duas partes, o que anatomicamente não é o ideal

· A parte descendente da serifa do travessão está separada da parte ascendente. A parte ascendente não faz sentido sem a descendente (o contrário [a descendente sem a ascendente] já é possível). Isto cria uma noção estranha de serifa partida, que não é muito lógica.



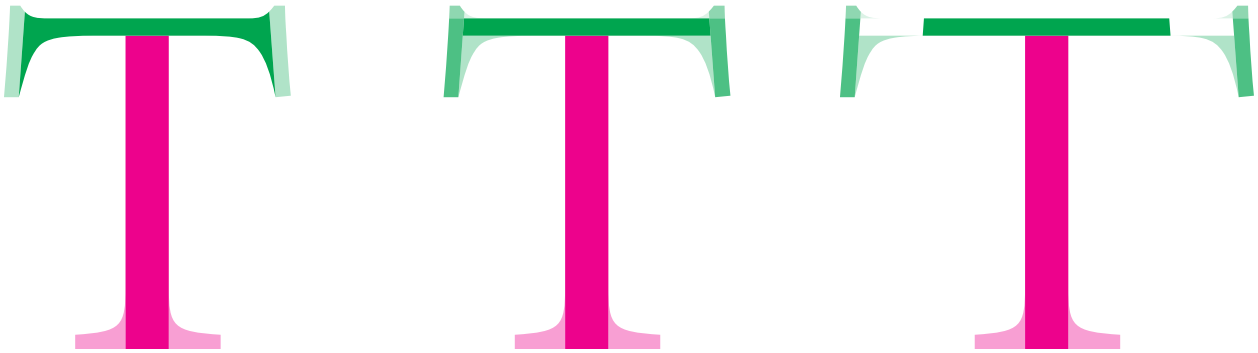
img 63
Corte perpendicular da serifas

Vantagens desta abordagem:

· Existe um travessão não separado.
· As serifas do travessão não estão separadas.

Desvantagens:

· O travessão fica encurtado, ou seja, precisa das serifas para ser percebido como um travessão. O que anatomicamente não faz muito sentido, porque a serifa é apenas uma terminação. Isto acontece mesmo quando o local de corte é afastado da zona automática (fim da curva).
· A haste termina um pouco a baixo, o que não é muito preocupante neste caso por se manter a sua principal dimensão vertical.



img 64 (esquerda)
Corte paralelo da serifas

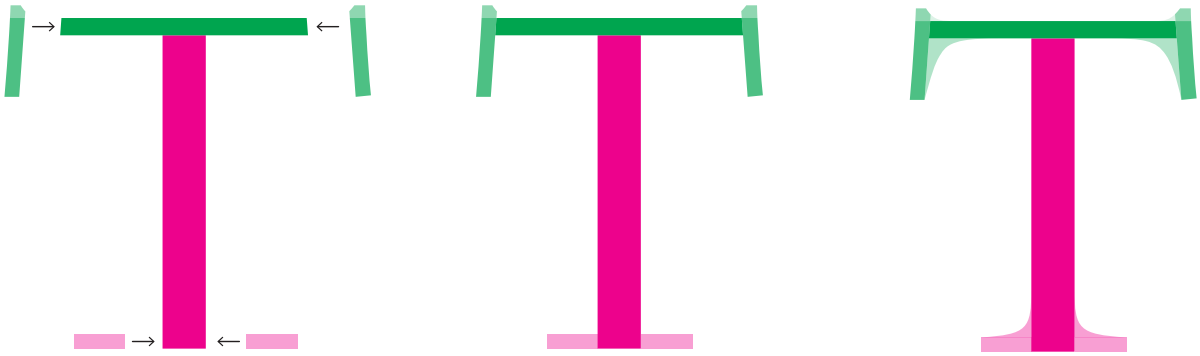
img 65 (centro & direita)
Corte paralelo da serifas com resolução por encaixe

(esquerda) Corte paralelo da serifas

Esta abordagem não faz sentido nenhum uma vez que deixa na mesma artefactos no travessão que, depois de tirar as serifas, teriam de ser tratados com serifas também. No entanto, é o ponto de partida para a próxima abordagem, ainda na mesma linha.

(centro e direita) Resolução por encaixe

Neste caso, a serifa é definida num intermédio entre as duas primeiras abordagens. o que resulta numa peça que “encaixa” no travessão no T. Isto não é ideal porque restringe a aplicação desta serifa a outros carateres. Teria de encaixar perfeitamente.



img 66
Funcionamento da abordagem da metamorfose

Abordagem da metamorfose

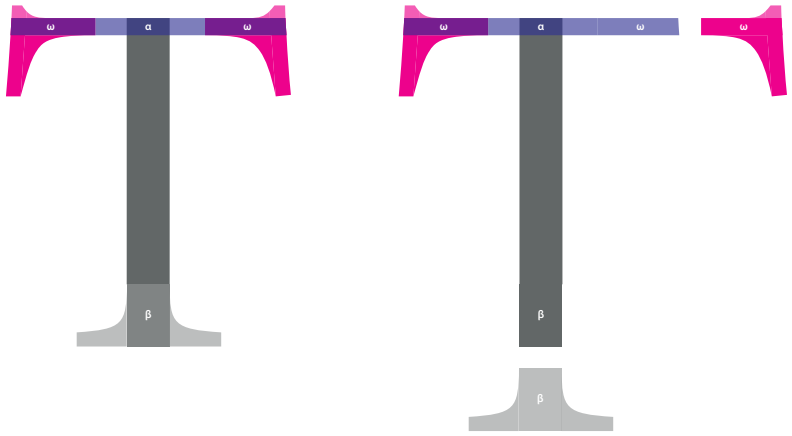
Esta abordagem é mais complexa porque trata de uma forma separada a junção da serifa com a parte anatómica onde se insere. Esta só faz parte da serifa quando esta é encaixada. Atenuando a forma da letra como se existisse uma tensão de superfície no contorno da forma. No entanto parece-me que será uma implementação mais dispendiosa do ponto de vista técnico. A parte ascendente da serifa continua a ser desprezável

Esta técnica resolve todos os problemas, gráficos e lógicos do ponto de vista anatómico. No entanto receio que seja de implementação mais complicada...

Trata-se de assumir que existem zonas de glifos que podem pertencer a mais do que uma parte anatómica. Isto, é até o mais lógico. Se virmos a documentação das partes anatómicas da letra (Amado, 2011) vê-se com frequência situações onde isto acontece. Por isso é que é necessário identificar apenas uma parte anatómica por glifo.

Exemplo: Para simplificar a explicação usei mais cores. No entanto toda a barra seria tratada como uma parte. Usando então a cor para a barra, e a escala de cinza para a haste:

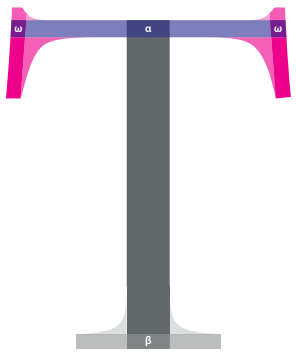
Aqui assume-se que as áreas α, β e ω não pertencem apenas a uma parte anatómica mas sim a várias. 6



img 67
Representação da sobreposição entre partes anatómicas.

Também aqui se pode utilizar a técnica da metamorfose

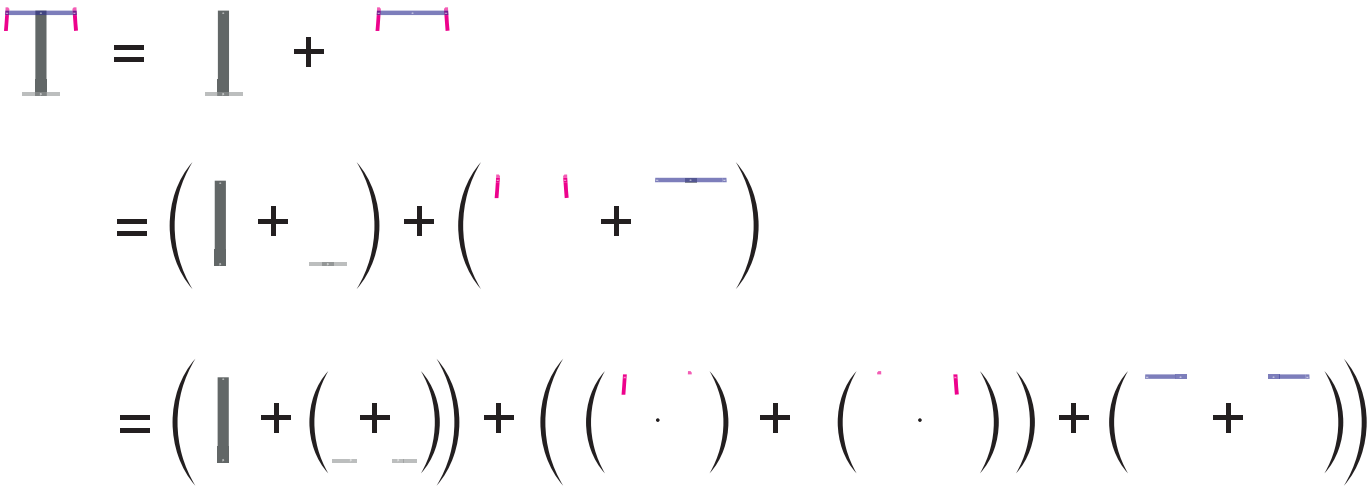
Dessa forma, a zona de sobreposição torna-se mais pequena porque a junta é tratada como um artefacto que simplesmente aparece depois de existir contacto.
Existe ainda a possibilidade para que a sobreposição possa acontecer a



img 68
Funcionamento da abordagem da metamorfose em sobreposição

vários níveis. Da seguinte forma (e ignorando as juntas para simplificar), por exemplo:

O ideal no nosso caso prático será utilizar uma abordagem mista entre zonas de sobreposição e justaposição. Uma vez que existem zonas que não



img 69
Representação das conexões que podem ser feitas a vários níveis de forma a obter uma letra através deste método

são necessariamente sobrepostas. Por exemplo as serifas da haste, podem ser tratadas como apenas uma serifa bi-lateral ou como duas serifas normais.

Tendo feito este estudo acerca de metodologias de corte de um glifo o próximo passo é a automatização deste processo. A ferramenta desenvolvida, teria a função de receber uma fonte e separar os seus glifos em partes anatómicas. Uma vez que se trata de um programa que terá de identificar, cortar e reconectar partes anatómicas o nome dado a esta ferramenta foi *Glyph Surgeon* (Cirurgião de Glifos).

No estudo anterior tornou-se obvio que os pontos pertinentes, para corte de partes anatómicas de forma tipograficamente lógica seriam: os pontos de desenho inicial da forma assim como os respetivos pontos de controlo, pontos intermédios e zonas de traço fino.

A linguagem de programação escolhida para a implementação foi Java com a utilização de Processing como uma biblioteca externa.

A lista de métodos a implementar teriam as seguintes funções:

- Importação da forma original dos glifos de uma font, contendo os seus pontos de desenho originais.
- Corte vectorial de uma forma e separação das partes resultantes.
- Deteção do esqueleto de uma forma (previamente implementado).
- Separação de um glifo de acordo com as estrutura detetada a partir da análise do seu esqueleto.

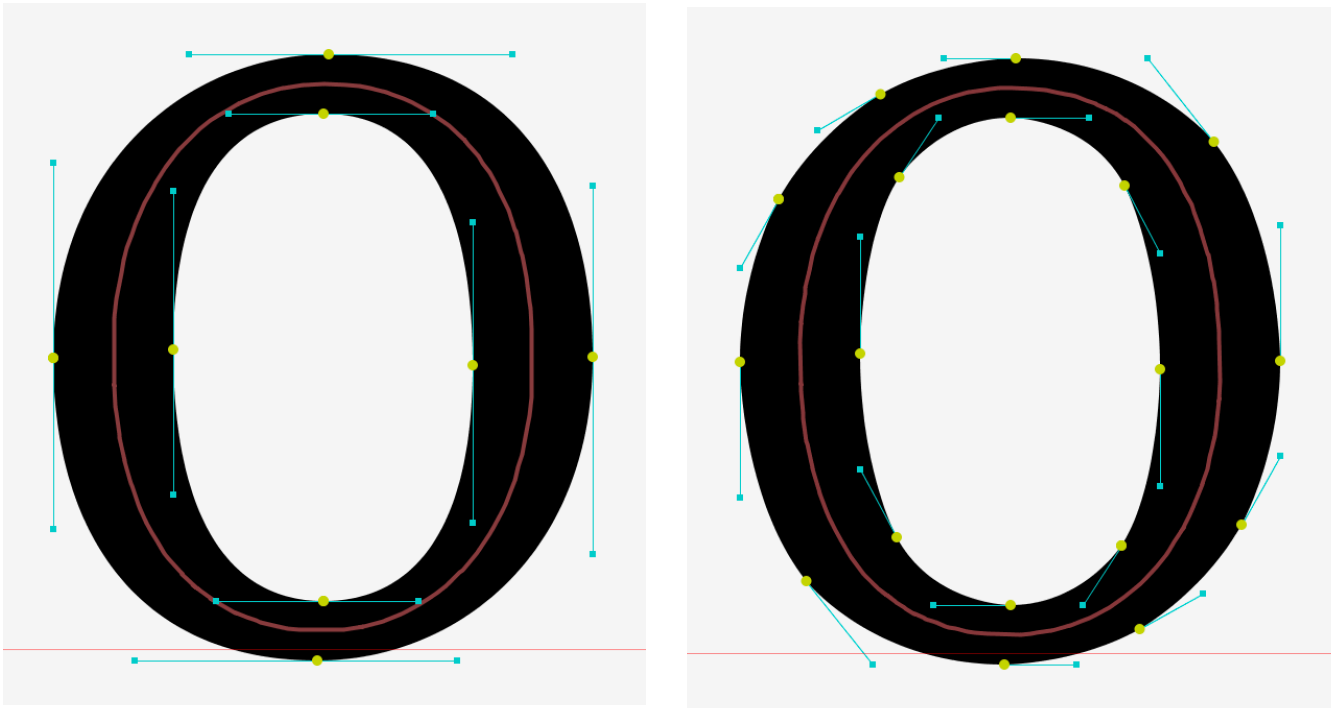
Para expandir as capacidades de desenho e manipulação de geometria foi utilizada a biblioteca geomerative (Marxer, n.d).

1.Importação de uma fonte

Quer o Processing quer a geomerative oferecem funções que permitem a importação das formas de glifos presentes em fontes. No entanto, apercebemo-nos que nenhuma das opções satisfazia as nossas necessidades completamente. O que era pretendido era aceder diretamente aos pontos que controlam o desenho do glifo, da mesma forma que é possível num programa de desenho vectorial, como o que foi utilizado no estudo de metodologias de corte.

A importação através do geomerative impedia a importação de ficheiros de fonte OTF permitindo apenas TTF. Isto, para além de representar logo à partida um corte no número de fontes que poderiam ser utilizadas com a ferramenta, apresenta outro problema que não era esperado. Os ficheiros TTF utilizam para representar as formas dos glifos curvas de bézier de nível 2 (quadráticas) ao contrário dos ficheiros OTF que utilizam

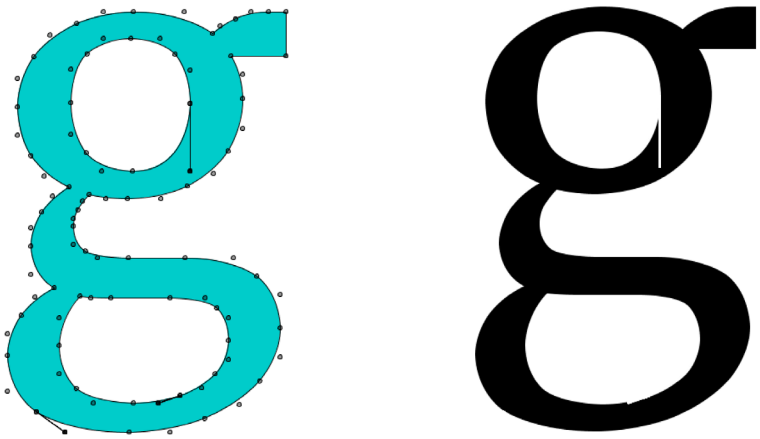
béziers de nível 3 (cúbicas). Como uma curva de bezier quadrática não tem a capacidade de representar curvas tão complexas quanto uma cúbica isto obriga a que sejam utilizados mais curvas para representar uma mesma forma (atualmente os programas de produção gráfica e de fontes utilizam béziers cúbicas para desenho. Caso um font seja guardada em formato TTF as curvas de bézier cúbicas são dissecadas em várias curvas de bézier quadráticas). A introdução de todos estes novos pontos na forma do glifo poderá tornar a tomada de decisão na altura do corte mais complexa e introduz ruído indesejável.



img 70
O mesmo desenho de uma letra o num ficheiro OTF (esqueda) e num TTF (direita). A forma OTF desenhada com béziers cúbicos tem 8 pontos + 16 controlos. A forma TTF com bézier quadráticos tem 18 pontos + 18 controlos.

No entanto, para além deste problema notou-se que o resultado da importação através do geomerative resultava na criação de alguns artefactos inesperados nos contornos curvos de alguns glifos. Em alguns pontos o contorno da forma é prolongado erradamente até ao ponto de controlo da curva.

A vantagem da utilização da importação através do geomerative era a capacidade de acesso direto aos pontos constituintes da forma e seus pontos de controlo.



img 71
Exemplo do problema notado na importação de fontes através da biblioteca *geomerative*

A importação através do Processing, permite a utilização de ficheiros de fontes quer em OTF (OpenType) quer em TTF (TrueType). No entanto um dos parâmetros da importação é o grau de simplificação da forma, ou seja, ao importar um forma pode-se especificar quão simplificado se quer o resultado, sendo a simplificação a transformação das curvas do glifo num conjunto maior ou menor de retas que aproximam a curva. Mas, ao tentar importar formas sem nenhum grau de simplificação de maneira a obter os pontos iniciais do desenho do glifo o resultado apresentado é completamente ilógico. Isto resulta no mesmo problema, se se optar simplesmente pela importação especificando um grau de detalhe muito elevado a quantidade de pontos obtidos introduzirá ruído indesejável na seleção dos pontos para corte mais tarde no processo e o desenho da forma deixará de ser uma curva para passar a ser uma aproximação poligonal.

img 72
Exemplo do problema notado na importação nativa do *Processing*.

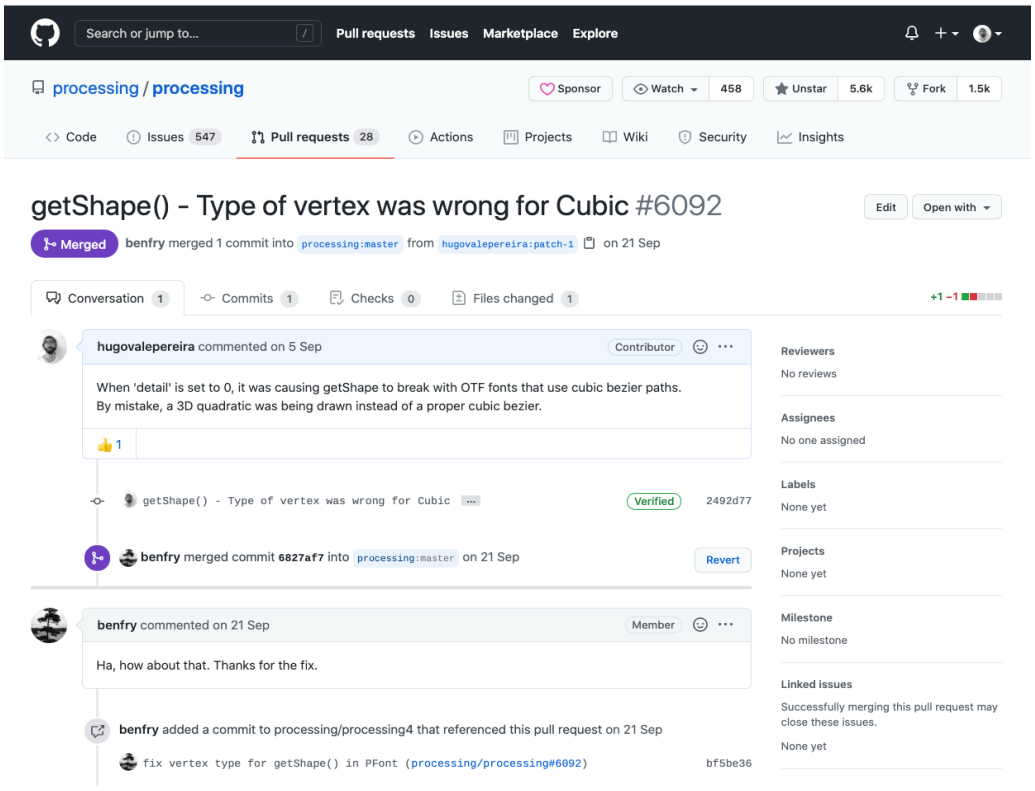
esq - grande nível de detalhe
centro - baixo nível de detalhe
dir - sem detalhe atribuído. Deveria desenhar segundo os pontos originais, no entanto apenas são desenhados alguns pontos

Estes dois problemas, por bloquearem o processo de trabalho motivaram a análise do código fonte das bibliotecas de forma a localizar os erros e a produzir uma abordagem própria mais eficaz.

A incapacidade de importação de ficheiros OTF por parte da *geomerative* acontecia devido à utilização, por parte da própria biblioteca, de uma outra dependência para a importação de fontes, a biblioteca *batikfont* que por si também não tem essa capacidade implementada.

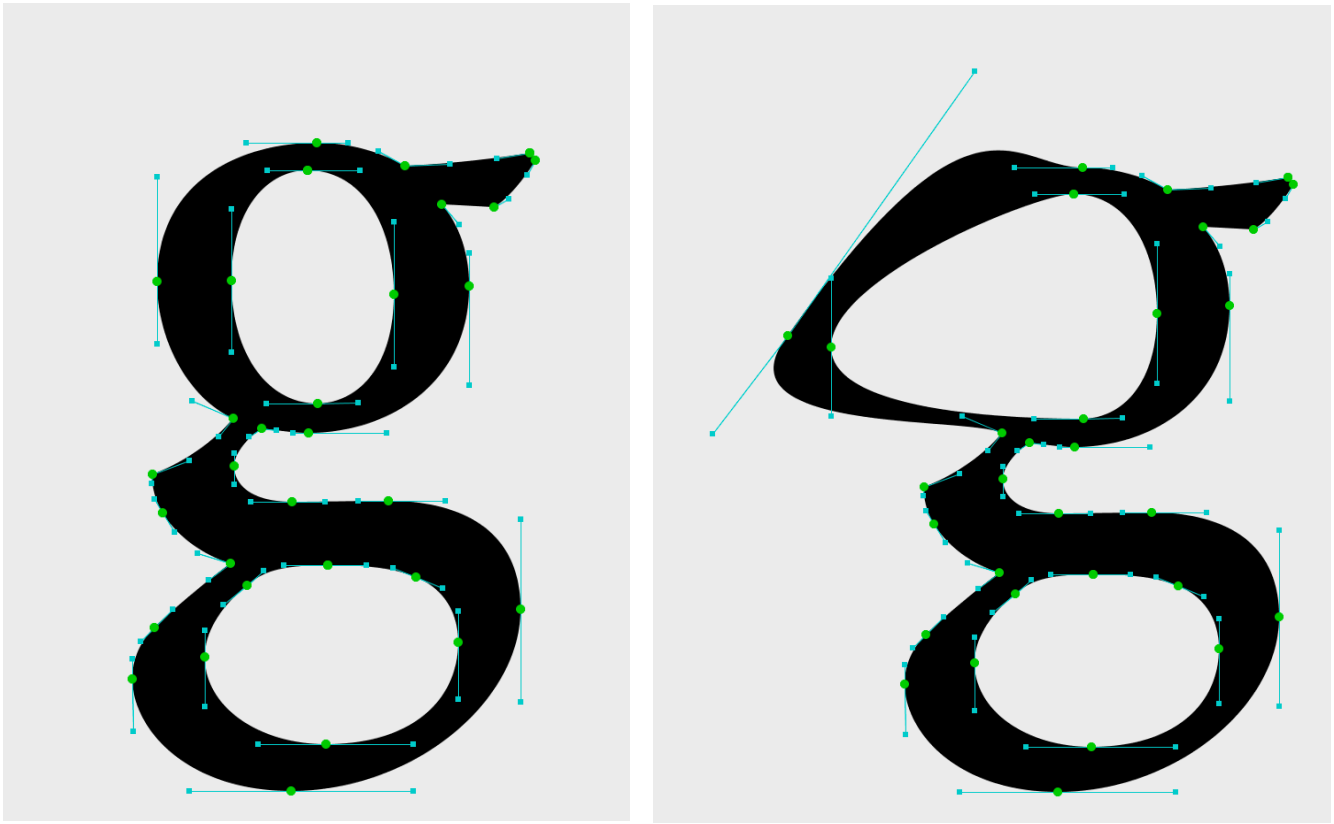
No entanto o erro de importação do Processing pôde ser identificado e corrigido. O problema só se reproduzia quando se utilizavam fontes OTF, precisamente porque a função que desenha curvas de bézier cúbicas tinha sido erradamente trocada por outra. Esta análise permitiu a implementação de um método de importação baseado no do Processing adaptado às necessidades específicas da ferramenta permitindo assim a organização dos pontos que definem a forma numa estrutura mais lógica, que separa em dois grupos os pontos de desenho dos pontos de controlo das curvas.

Uma vez que o Processing se trata de uma biblioteca *opensource* em constante desenvolvimento colaborativo, foi proposta uma sugestão de correção do erro no repositório online da plataforma, que foi posteriormente aceite.



img 73
Pull Request no repositório online do *Processing*, com a sugestão de correção do erro encontrado.

Desta forma, o problema de importação de um ficheiro de fonte está resolvido permitindo a manipulação direta dos pontos que formam o desenho de um glifo.



img 74
Utilização do método desenvolvido para manipulação dos pontos que definem o desenho de um glifo.

2. Corte Vectorial de uma forma

Depois de ter aceso os pontos constituintes de uma forma, a próxima necessidade é a de cortar a forma através de uma linha de corte. Este é o tipo de processos que bibliotecas de manipulação de geometria com a *geomerative* ajudam a executar. Por isso a biblioteca foi estudada de forma a selecionar o método correto para corte de uma forma. No entanto, o desejado, não existe.

Sem qualquer dúvida, esta foi a parte mais dispendiosa de todo o projeto uma vez, que não foi encontrado publicamente nenhum algoritmo documentado ou biblioteca que executasse o que pretendíamos na totalidade. Assim sendo apenas com acesso a definições matemáticas e um processo de tentativa e erro, o método pertendido de corte foi implementado.

O pretendido:

Tendo uma forma geométrica definida por pontos (no nosso caso um glifo), e tendo dois pontos que definem uma linha de corte. Ao executar uma determinada função o resultado deveria ser as várias partes que são separadas pela linha, definidas também elas vectorialmente por pontos.

Um processo semelhante ao utilizado nos vários programas de desenho gráfico vectorial (Ex: Adobe Illustrator).

O que existia:

A biblioteca geomerative oferecia funções capazes de separar formas em partes mas nenhuma que fosse capaz de resolver o problema que propúnhamos. Todos os métodos de corte de curvas passavam sempre pela simplificação das curva num conjunto aproximado de retas.

Sabendo que o primeiro passo para cortar uma forma com uma linha é calcular as intersecções entre estas tentou-se procurar funções que tivessem essa capacidade.

Calcular os pontos de intersecção entre uma linha e uma forma era possível, no entanto, a informação resultante (coordenadas do ponto de intersecção), não era suficiente para introduzir novos pontos da forma precisamente no local onde a intersecção tinha sido detetada. Isto acontece porque o método pelo qual é possível introduzir um novo ponto numa forma é fornecendo um valor entre 0.0 e 1.0, que representa o avanço de um ponto no *outline* da forma, sendo 0 o início e 1 o final.

A separação de uma forma em duas é possível, mas também apenas utilizando um valor entre 0.0 e 1.0 e não com valores de coordenadas. De qualquer forma o resultado deste tipo de separação é inútil uma fez que retorna simplesmente o traçado da forma até esse determinado ponto e conectando-o de novo ao primeiro ponto da forma.

Operações entre duas formas eram também possíveis, no entanto as curvas eram primeiro simplificadas em linhas resultando em formas pouco fidedignas.

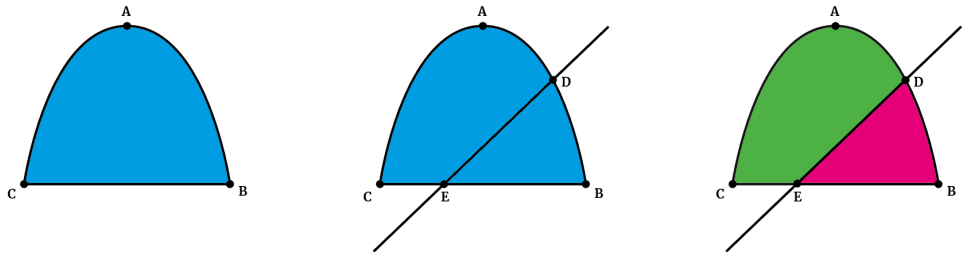
Tornou-se então óbvio que teríamos de fazer a investigação necessária e desenvolver nós os algoritmos capazes de fazer este tipo de calculo.

O Processo :

Para cortar uma forma geométrica o primeiro passo é o de calcular as

intersecções que uma linha cortante faz nas linhas que definem a forma.

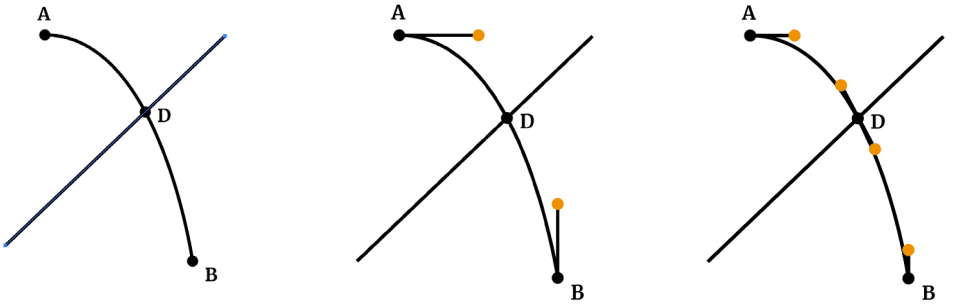
Depois, tendo acesso aos pontos de intersecção as novas formas têm de ser separadas uma da outra e depois fechadas.



img 75
Corte de uma figura geométrica através do calculo das intersecções entre uma reta e os seus traços.

Por exemplo, tendo uma forma definida pelo conjunto de pontos [A,B,C], primeiro terá de se encontrar os pontos D e E, que representam as intersecções da linha com a forma e depois criar duas novas formas que serão definidas pelos conjuntos [A,D,E,C] e [D,B,E].

Para calcular os pontos de intersecção da linha com a forma é preciso calcular os pontos de intersecção entre a linha e cada um dos seus segmentos. No caso da intersecção entre duas linhas, o caso é relativamente trivial podendo ser calculando o ponto de intersecção utilizando as equações pelas quais elas podem ser definidas, igualando e resolvendo. Este problema está resolvido pela *geomerative*, e existem muitos recursos *opensource* que o resolvem, logo não ocupou muito tempo de trabalho. No entanto se a intersecção acontecer entre a linha e uma curva o problema é mais complexo.



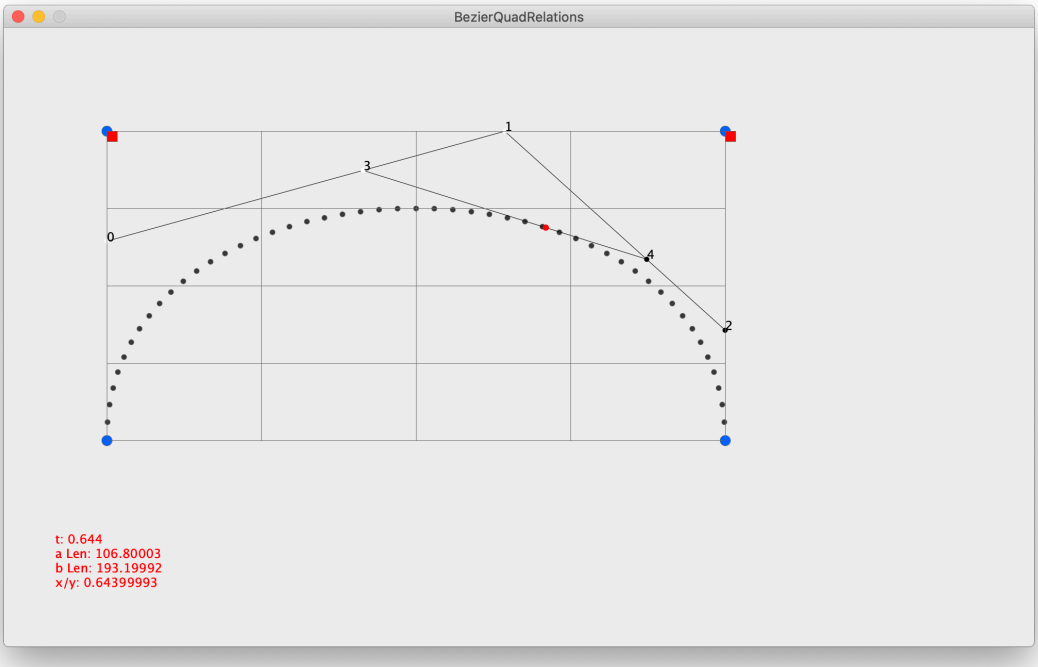
img 76
Intersecção e secção de uma curva de bézier por uma reta.

Considerando desta vez o segmento [A,B] da forma previamente utilizada e a mesma linha cortante. A curva [A,B] é calculada utilizando dois pontos de controlo que a definem (marcados a laranja), estes pontos definem em conjunto com os pontos A e B uma curva de bézier cúbica. Neste caso não é apenas necessário encontrar a posição do ponto D mas também o reposicionamento dos pontos de controlo de forma a que as duas curvas resultantes mantenham a forma inicial.

Por isso o primeiro desafio é precisamente saber como se pode dividir uma curva de bézier.

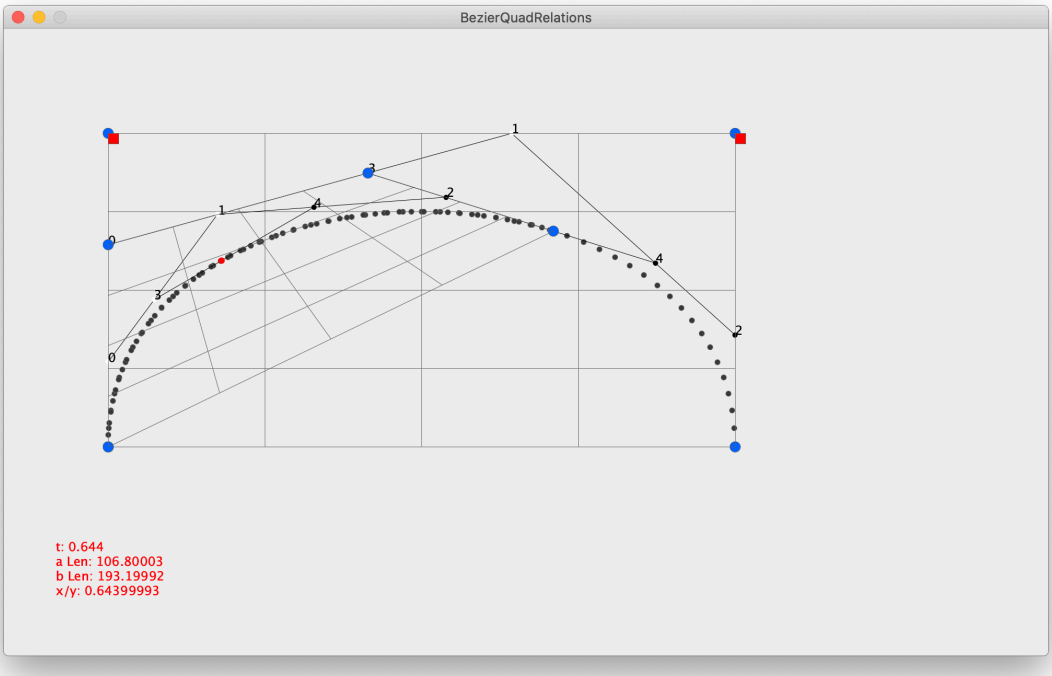
Um curva de bézier é desenhada através de um processo iterativo para valores de avanço entre 0 e 1, sendo 0 o início da curva e 1 o fim. Primeiro, criam-se linhas entre os seus pontos de controlo, o resultado são três retas.

Para calcular as coordenadas de um ponto pertencente à curva para um avanço, digamos, *t* calcula-se a interpolação com o valor *t* para cada uma das retas criadas anteriormente. Esses novos 3 pontos são agora eles unidos criando agora 2 retas e o processo é repetido, calculando a interpolação para o valor *t* para cada uma das novas retas. Disto resultam 2 novos pontos que são também unidos, e para a reta criada também o valor da interpolação para *t* é calculado. Esse ponto representa o ponto da curva de bézier para o valor *t*. Para desenhar toda a curva é preciso calcular continuamente os valores entre 0 e 1 através deste método.



img 77
Curva de bézier e pontos utilizados para o seu cálculo

Para estudar as curvas um pequeno programa foi criado onde este processo pode ser visualizado. Assim empiricamente notou-se que estes mesmos pontos que são criados ao construir uma curva de bézier são precisamente os pontos de controlo que podem ser utilizados na divisão de uma curva em duas novas curvas dado um determinado ponto de avanço.



img 78
Uma segunda curva de bézier inserida na curva inicial

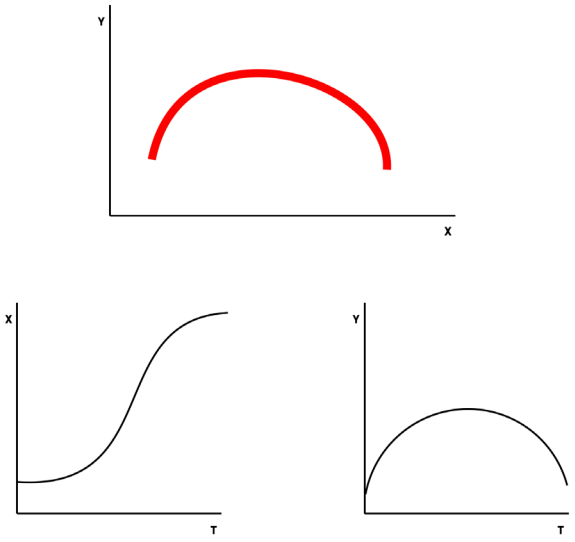
A questão é agora como obter esse valor representativo do avanço para uma intersecção com uma reta.

Outra forma de definir uma curva de bézier alternativa ao algoritmo apresentado anteriormente é matematicamente através de uma equação polinomial:

$$B(t) = (1 - t)^3 \cdot B_0 + 3t\,(1 - t)^2 \cdot B_1 + 3t^2\,(1 - t)^2 \cdot B_2 + t^3 \cdot B_3,\ t \in [0,1]$$

Em que B(t) representa os pontos (Bx,By) da curva para um determinado avanço *t* compreendido entre 0 e 1. E onde B_0, B_1, B_2 e B_3, representam os pontos de controlo da curva.

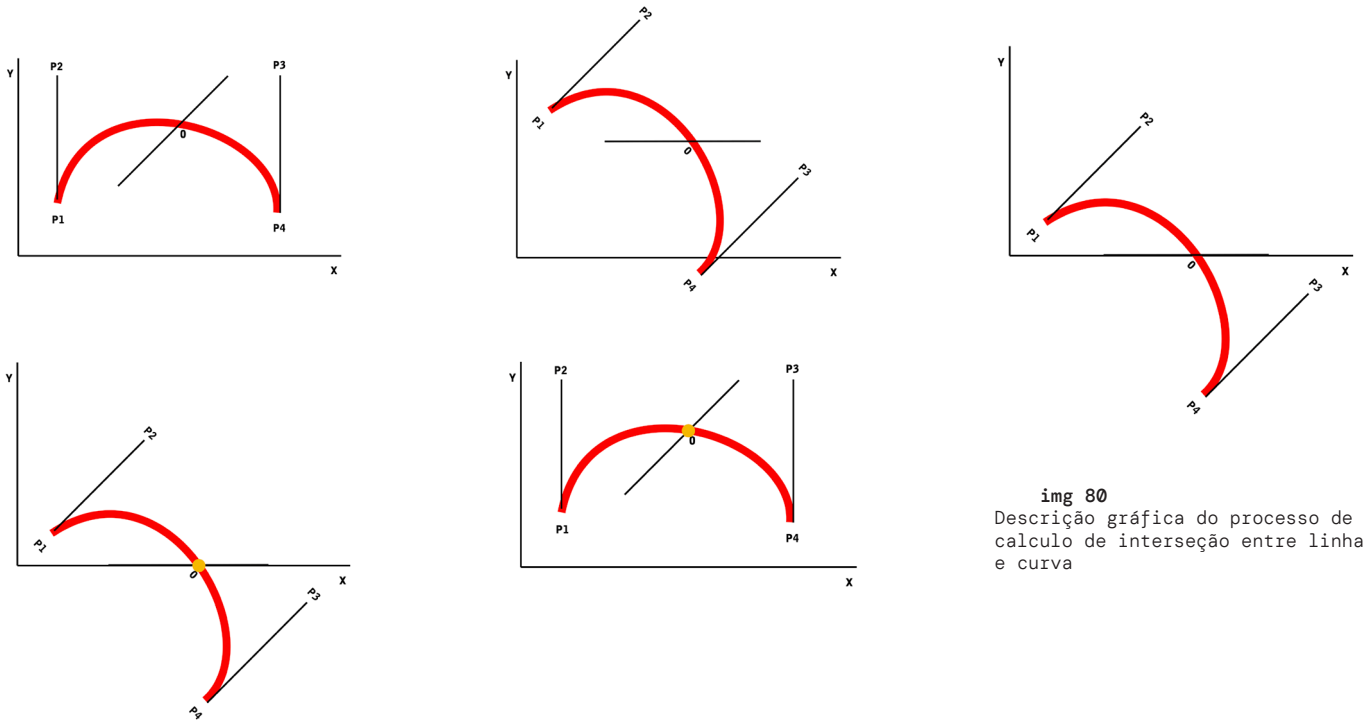
Ou seja, os pontos da curva de bézier são definidos através de uma função polinomial que representa a variação do valor da coordenada horizontal do ponto, e outra para a coordenada vertical.



img 79
A representação da variação dos valores de X e Y de uma curva de bézier (a vermelho) em função do valor *t*

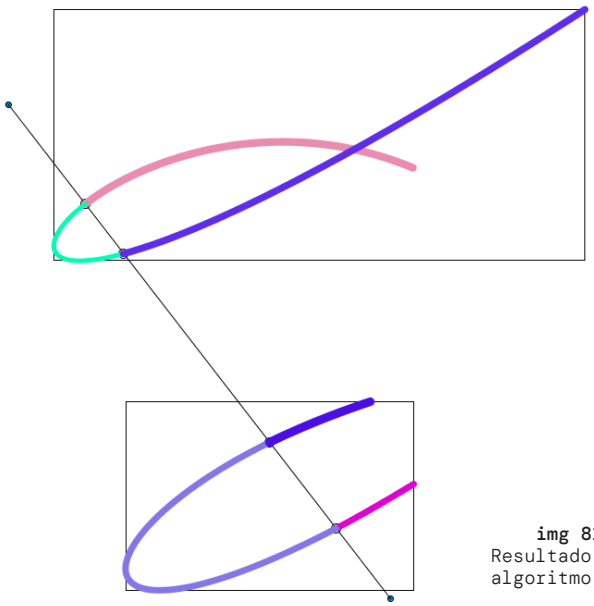
Desta forma, para calcular a intersecção entre uma curva e linha cortante um método possível pode ser descrito da seguinte forma:

- Tendo uma curva de bézier num referencial e uma linha que o intersecta.
- Aplica-se uma transformação geométrica de forma a alinhar a linha cortante com o eixo horizontal.
- Ao ter a curva nesta posição, calculam-se as raízes do polinómio (valores para os quais y=0) para a função representativa dos valores de y, e obtém-se assim os valores do avanço para os quais a curva intersecta o eixo horizontal (que como a curva foi rodada representa a linha cortante).
- Se estes valores forem compreendidos entre 0 e 1, podem ser introduzidos na função inicial que representa a curva e obter o ponto de intersecção. (Kamermans, n.d.)



img 80
Descrição gráfica do processo de calculo de interseção entre linha e curva

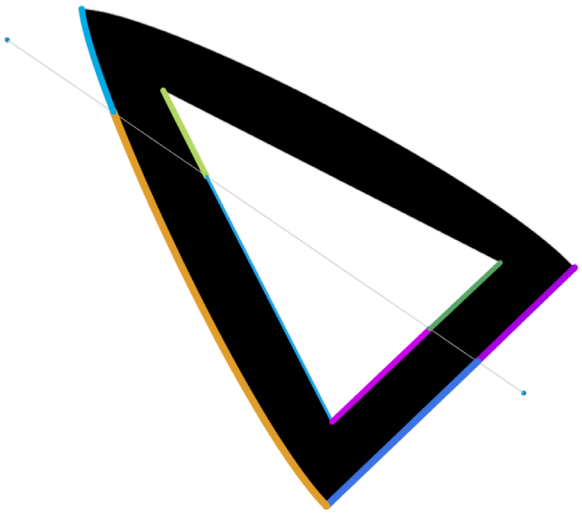
Este algoritmo foi implementado com sucesso, conseguindo identificar os pontos de intersecção direta entre uma linha e uma curva de bézier.



img 81
Resultado da implementação do algoritmo de corte de curvas

Neste exemplo o algoritmo deteta inicialmente se existe a possibilidade de intersecção, calculando a intersecção com o rectângulo envolvente da curva. E se essa possibilidade existir calcula os pontos de intersecção e depois desenha com diferentes cores as curvas resultantes.

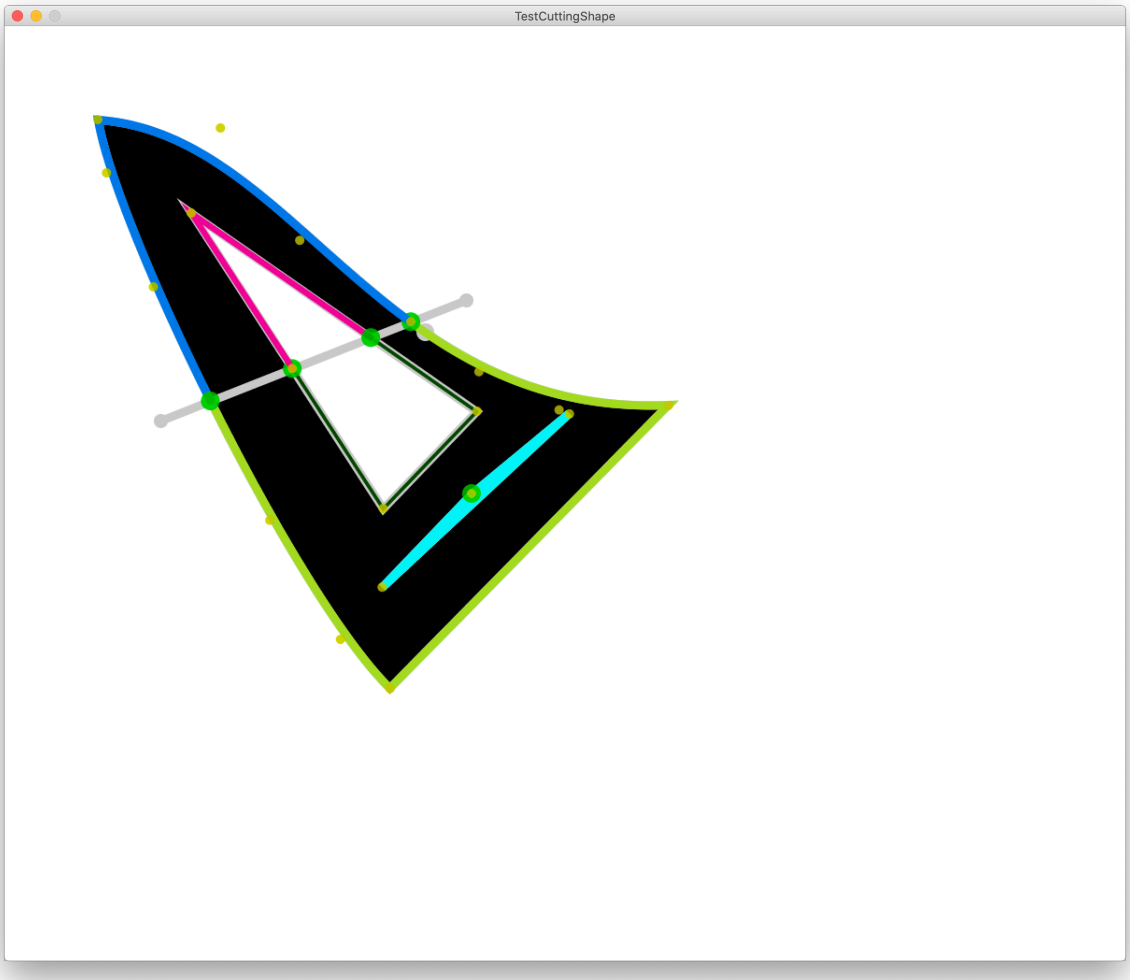
Este algoritmo foi por sua vez aplicado a formas calculando as interseções entre uma linha cortante e cada uma das curvas que definem a forma.



img 82
Resultado da implementação do algoritmo de corte de curvas em formas

O passo seguinte é efetivamente separar as duas partes e fechar as formas resultantes.

Para isso é preciso primeiro agrupar as curvas resultantes em grupos, bastando para isso marcar como sendo do mesmo grupo curvas que ao longo do percorrer do contorno estão entre pontos de intersecção da linha com a forma.

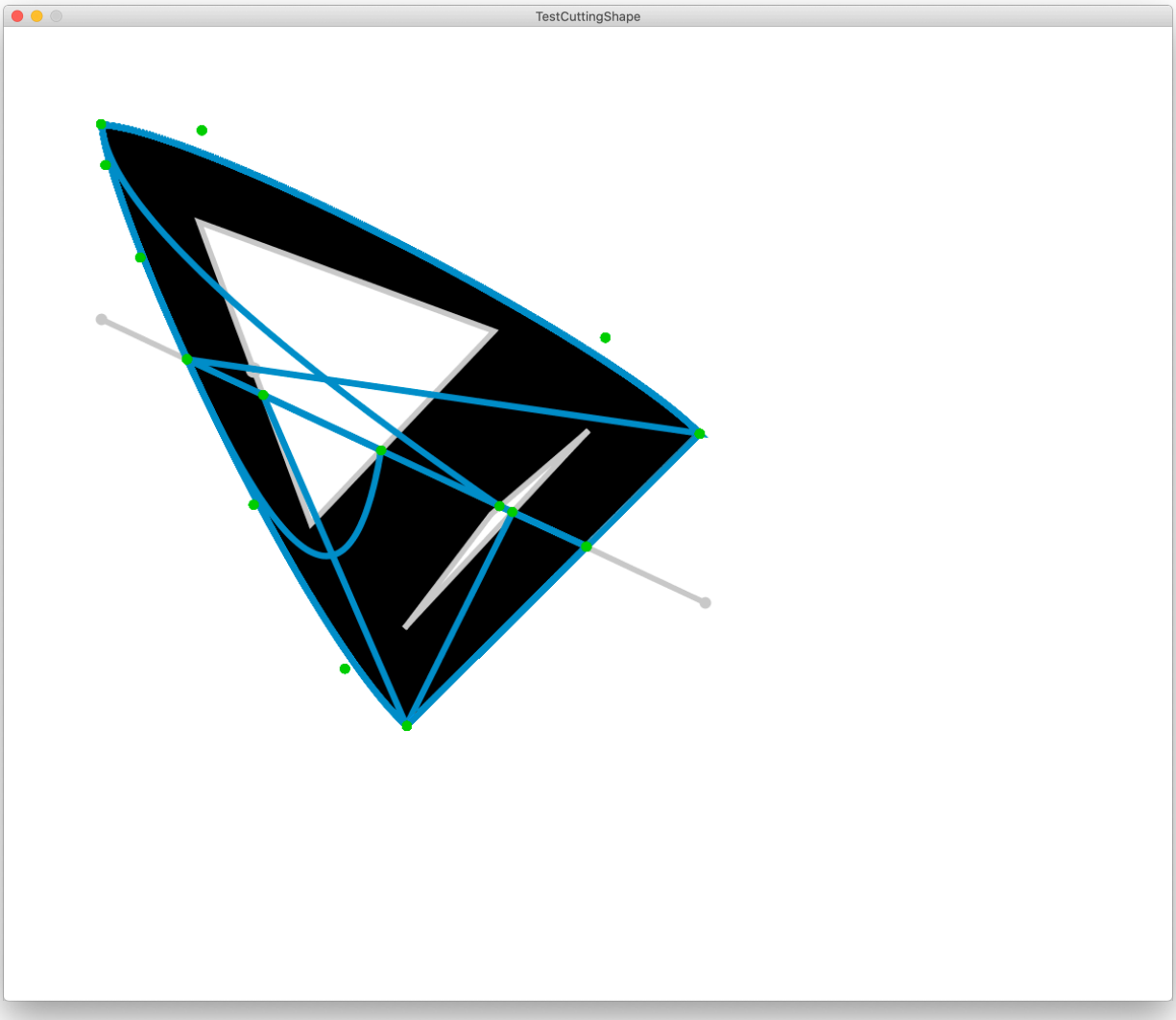


img 83
Resultado da implementação do agrupamento de linhas

É preciso depois fechar cada uma das formas resultante ao longo da linha cortante. Para isso é preciso primeiro considerar todos os casos de corte possíveis. Por exemplo, se a forma tiver uma contra-forma no seu interior, como por exemplo um *A*, o resultado do corte não serão duas formas, mas sim apenas uma forma com um novo contorno, uma vez que o interior da forma continuará conectado.

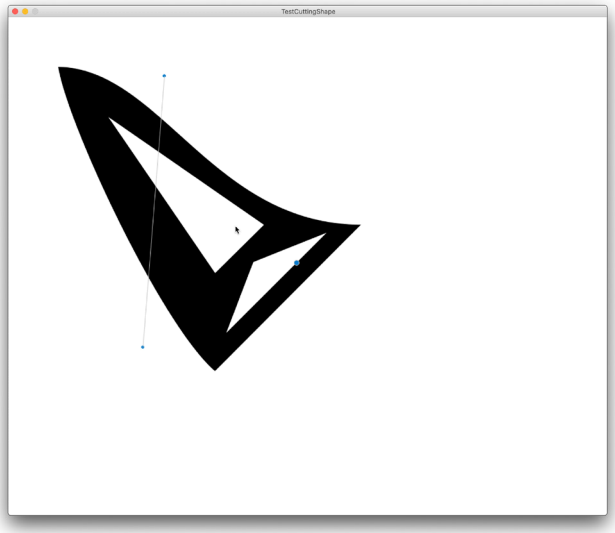
Para resolver este problema é necessário percorrer o caminho da forma naturalmente pela direção nativa da forma. Isto é, num sentido por fora, e no sentido contrário nas contra-formas. Assim, ao percorrer a forma, se se encontrar um ponto de intersecção, o troço da linha cortante até à próxima intersecção é adicionado à nova forma. Este processo é realizado até se retornar ao ponto inicial e todos os caminhos da forma terem sido percorridos.

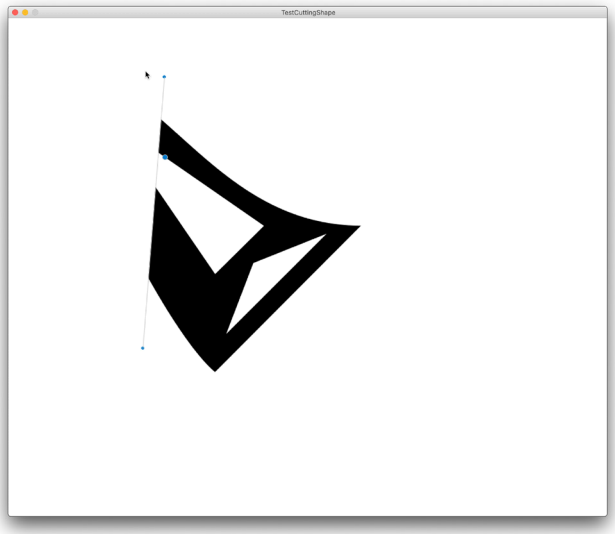
A complexidade deste algoritmo, ocupou bastante tempo de implementação e o processo foi guiado por um método de tentativa e erro à medida que muitos problemas eram resolvidos e novos problemas surgiam.



img 84
Erro na tentativa de implementação o fechamento dos caminhos

O resultado final da implementação deste algoritmo é robusto o suficiente para a implementação do pretendido no corte das partes anatómicas de glifos. Por isso não se resolveram alguns problemas menores que não seriam decisivos para este projecto em específico, mas que seriam caso a aplicação do algoritmo de corte fosse mais generalizada. Como por exemplo, o algoritmo falhar se for feito um corte através de duas contra-formas e a linha terminar numa delas.





img 85 (topo)
Posicionamento da linha na forma

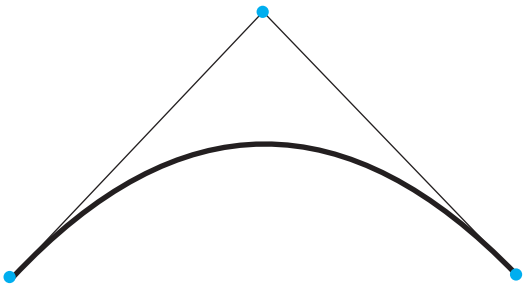
img 86 (baixo)
Resultado do corte

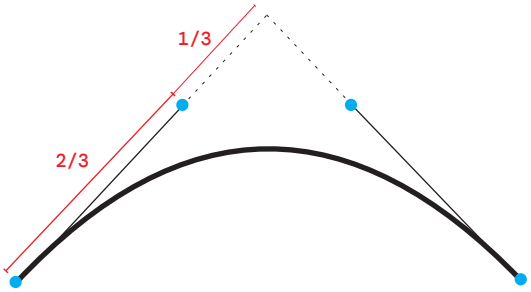
Outra abordagem interessante seria a de calcular as intersecções entre duas curvas de forma a permitir cortes não apenas feitos por retas.

De qualquer forma, o trabalho desenvolvido representa um grande valor acrescentado para qualquer biblioteca de geometria que recorra a simplificação de curvas a polígonos para cálculos de intersecções. Como é o caso da biblioteca mais completa para Processing, *geomerative*.

Com o conteúdo estudado acerca curvas de Bézier, propôs-se resolver um problema que solucionaria o excesso de pontos numa fonte TTF convertendo o seu contorno num compativel com OTF. Isto é útil no nosso contexto uma vez que homogeniza o tipo problema que a ferramenta tem de resolver, como também aumenta o número de ficheiros de fonte que podem ser aceites pela ferramenta.

Transformar uma curva de bézier quadrática numa cúbica é relativamente fácil (o contrário, já não é verdade uma vez que todas as quadráticas podem ser representadas por cúbicas mas apenas uma pequena parte das cúbicas podem ser representadas por quadráticas) e está bem documentado qual o método online. Basta simplesmente colocar os pontos de controlo da curva cúbica a 2/3 da distância entre os pontos inicial e final da curva e o único ponto de controlo da curva quadrática. (Kamermans, n.d.)



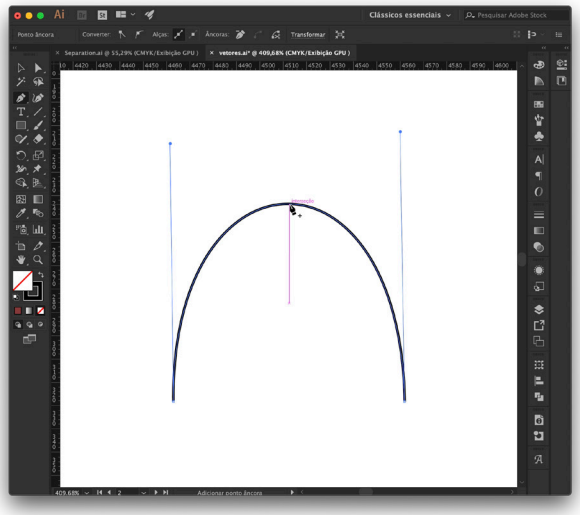


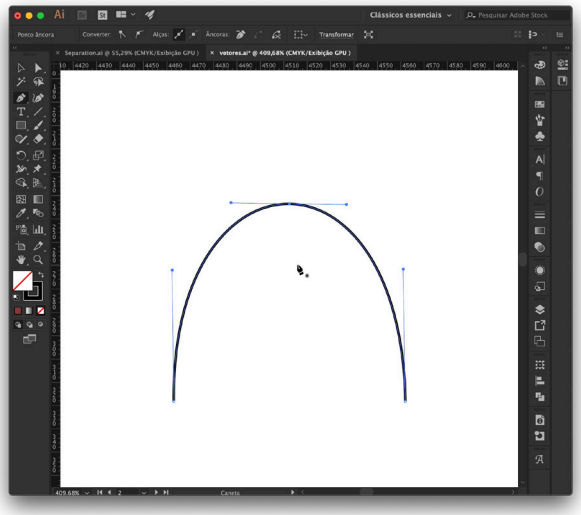
img 87 (esquerda)
Representação de uma curva de bézier quadrática (um ponto de control)

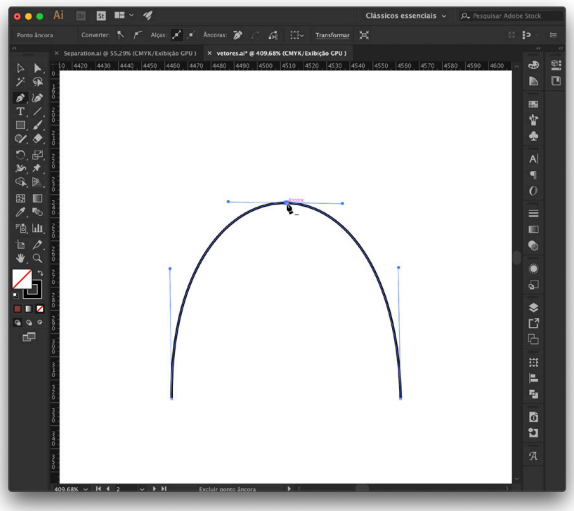
img 88 (direita)
Representação da mesma curva através de um bézier cúbico (dois pontos de controlo)

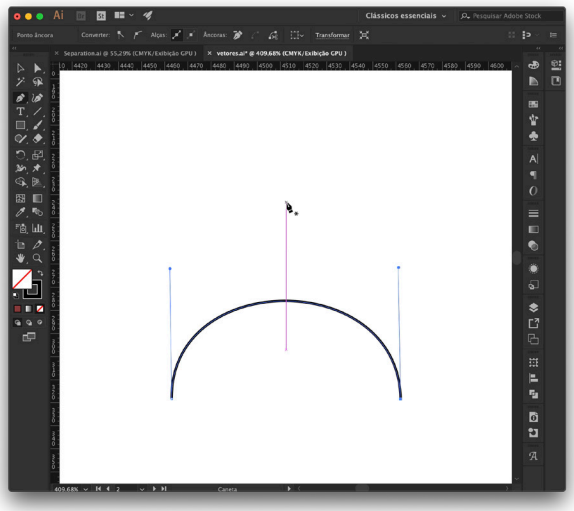
Esta transformação, de quadrática é cúbica, por si só não oferece grande vantagem uma vez que o número de pontos da curva não se modifica e ainda é acrescentado um ponto de controlo. No entanto, a transposição de todas as curvas para curvas de bézier cúbicas é um bom primeiro passo para a simplificação.

Empiricamente, por experiência de utilização de programas de desenho vectorial como é o caso do Adobe Ilustrator (que utiliza curvas de bézier cúbicas para representar curvas) é possível comprovar que em alguns casos, é possível transformar duas curvas de bézier cúbicas em apenas uma. A imagem seguinte demonstra isso mesmo.









img 89 (topo esquerda)
Se no Adobe Illustrator tivermos uma curva, é possível adicionar novos pontos em qualquer ponto no caminho.

img 90 (topo direita)
Depois de adicionar um novo ponto a curva mantém-se igual. Mas sendo agora na prática definida por duas curvas que são definidas por novos pontos de controlo.

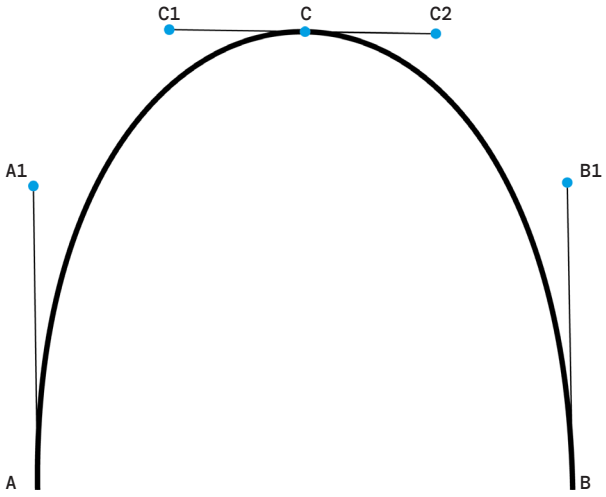
img 91 (baixo esquerda)
Este novo ponto pode também ser eliminado do caminho.

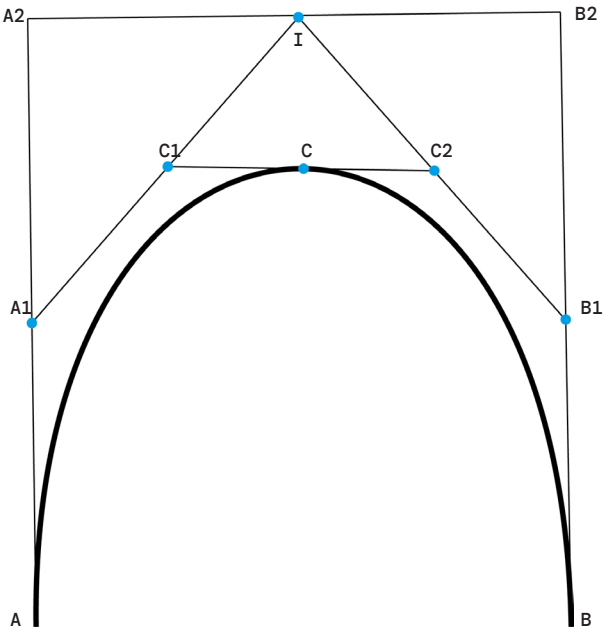
img 92 (baixo direita)
No entanto o resultado não é a curva inicial, mas sim uma curva descrita apenas pelos pontos de controlo não adjacentes ao ponto eliminado. Não existe forma de eliminar o ponto mantendo a forma da curva.

Embora o resultado final seja diferente do inicial, é possível dizer que neste caso, como sabemos que inicialmente assim era, deveria ser possível remover o ponto mantendo a forma da curva. Ou seja, simplificar duas curvas de bézier em apenas uma. Desta forma podemos afirmar que qualquer conjunto de curvas de bézier cúbicas que tenham inicialmente sido definidas por apenas uma curva de bézier têm de poder ser redefinidas por apenas uma curva. A questão aqui é saber, se um conjunto de curvas pode ou não ser simplificado. No desenvolver do projeto, foi possível definir um método capaz de fazer essa distinção e que pode ser descrito da seguinte forma.

O método desenvolvido analisa apenas pares de curvas. Isto não é de forma nenhuma limitador do grau de simplificação que permite, uma vez que depois de simplificado um par, este pode ser analisado como uma nova curva em conjunto com outra, formando um novo par.

O método de desenho de uma curva de bézier, como exposto previamente pode ser utilizado para procurar os pontos de controlo necessários para desenhar duas curvas com a mesma forma da inicial. Desta forma sabemos que se um par de curvas "encaixar" nos traços de desenho de uma curva, esse par pode ser simplificado por essa curva.





img 93 (esquerda)
Representação de duas curvas de Bézier que poderiam ser simplificadas

img 94 (direita)
As mesmas curvas integradas nas linhas necessárias para desenhar uma curva de bézier.

Se a curva for simplificável, sabemos que o ponto C se encontra a uma distância de C1 que relativa à distância de C1 a C2 representa o valor de avanço *t* que é utilizado também em A-A1 e B-B1. Assim, retirando esse valor do distanciamento relativo do C em relação a C1, podemos usar esse valor para calcular quais as posições de A2 e B2 que são unidos. Assim, se se prolongar as retas formadas por A1-C1 e B1-C2 pode-se calcular a sua intersecção.

Se esse ponto de intersecção for coincidente com a reta A2-B2 e a sua distancia relativa a A2 em relação a A2-B2 for igual a *t*, a curva pode ser simplificada e descrita por uma única curva de bézier cúbica com os pontos de controlo A2 e B2.

Assim, este algoritmo pode ser utilizado para simplificar glifos de fontes em TTF.

Esqueleto

Tendo agora a capacidade de cortar qualquer forma vectorialmente. A tarefa seguinte é escolher a localização dos pontos de corte. Para isso a integração da deteção de esqueletos de formas teve de ser incorporada.

Durante o decorrer do desenvolvimento, um novo algoritmos de deteção de esqueletos foi encontrado. Este novo algoritmo tem a vantagem de funcionar com base em cálculo vectorial ao invés dos algoritmos utilizados previamente, o que permite a utilização em qualquer tamanho, algo que era a fragilidade dos algoritmos implementados previamente.

Ao invés vez de implementar o algoritmos de raiz uma biblioteca que já tinha o algoritmo implementado foi utilizada *FLOMAT (*Steenkamp, n.d.). Para além da deteção de esqueleto vectorialmente, a biblioteca também oferecia simplificação de esqueleto, e funcionava com uma prestação superior aos algoritmos previamente implementados. No entanto como se tratava de uma biblioteca implementada em node.js e não em Java, a integração direta no programa já desenvolvido não seria possível.

Por isso, foi desenvolvida uma pequena API em node.js.

Esta API simplesmente como ponte entre as duas linguagens. O programa principal gera uma String SVG que descreve a forma, e envia essa informação através de um pedido HTTP para o enderço onde a API está hospedada. A API recebe o pedido HTTP e forneçe a informação geométrica às funções da biblioteca *FloMat* que devolvem um Objecto JavaScript que descreve a forma do esqueleto e os pontos onde existem junções. Um JSON com dois arrays (um que descreve os traços do

esqueleto, e outro com as coordenadas das junções) é devolvido como resposta do pedido HTTP.

Uma das grandes vantagens da utilização deste método de deteção de esqueletos é a capacidade definir o grau de detalhe que pretendemos. Na implementação da biblioteca *FloMat* os valores de detalhe mais baixos produzem esqueletos mais detalhados, enquanto quanto mais alto o valor, mais simplificada é a forma do esqueleto da forma. Isto é útil, porque permite uma melhor adaptação entre diferentes tipos de letra. Nos teste utilizados, foram utilizados dois valores (1,5) que resulta em esqueletos com detalhe suficiente para identificar todas as partes anatómicas individualmente, e (7,0) que resulta num esqueleto mais simplificado que ignora partes anatómicas de menor escala, como por exemplo serifas.

img 95 (topo esquerda)
Esqueleto gerado com nível de detalhe 0.0. Os "ossos" no esqueleto prolongam-se até todos os vértices da forma.

img 96 (topo direita)
Esqueleto gerado com nível de detalhe 1.5. Os "ossos" do esqueleto representam individualmente cada uma das partes anatómicas do glifo.

img 97 (baixo esquerda)
Esqueleto gerado com uma valor de detalhe de 7.0. Os "ossos" apenas representam o desenho das partes anatómicas mais marcantes do glifo.

Tendo acesso aos vários segmentos do esqueleto e aos pontos onde as suas partes se juntam é necessário agora analisar os seus traços e tentar perceber em cada junção qual é o osso que forma um contínuo e qual o osso que termina na junção e deve ser separado. Desta forma, localizar zonas de quebra e associar partes que poderiam ser consideradas apenas uma.

img 98 (esquerda)
Traços (a vermelho) e pontos (a amarelo) de junção resultantes do processo de deteção de esqueleto.

img 99 (centro)
Se simplesmente os traços do esqueleto forem cortados nos pontos, o resultado será um número maior de ossos representantes de partes anatómicas do que o esperado. Pode ver-se por exemplo que a parte roxa, também deveria ser identificada como parte da haste (a vermelho), no entanto é identificada como uma outra parte separada.

img 100 (direita)
Resultado pretendido. Agrupando em cada ponto os traços que devem representar um traço contínuo.

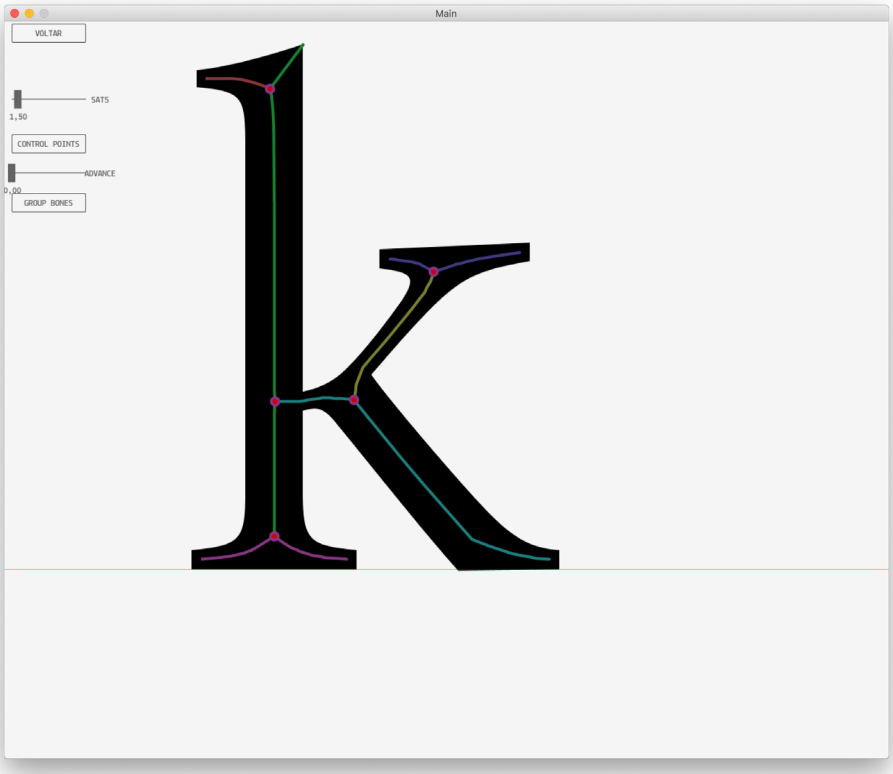
Este resultado foi obtido implementando um algoritmo muito simples que analisa os ângulos formados entre os traços. Normalmente, quando existe uma junção, esta é composta por 3 ramos, e na maior parte das vezes dois desse ramos representam na realidade apenas um onde um segundo é conectado. A informação que temos do esqueleto não engloba nativamente estes conceitos de agrupamento entre os vários traços. Por isso é necessário fazer esse calculo para obter um desenho do esqueleto que melhor represente as partes anatómicas de um glifo.

img 101 (esquerda)
1. Cada junção é identificada. São calculados os valores dos ângulos formados entre os traços.

img 102 (centro)
2. É identificado o maior ângulo.

img 103 (direita)
3. Os dois traços adjacentes a esse maior ângulo são identificados como apenas um e agrupados. O traço restante é considerado um novo "osso".

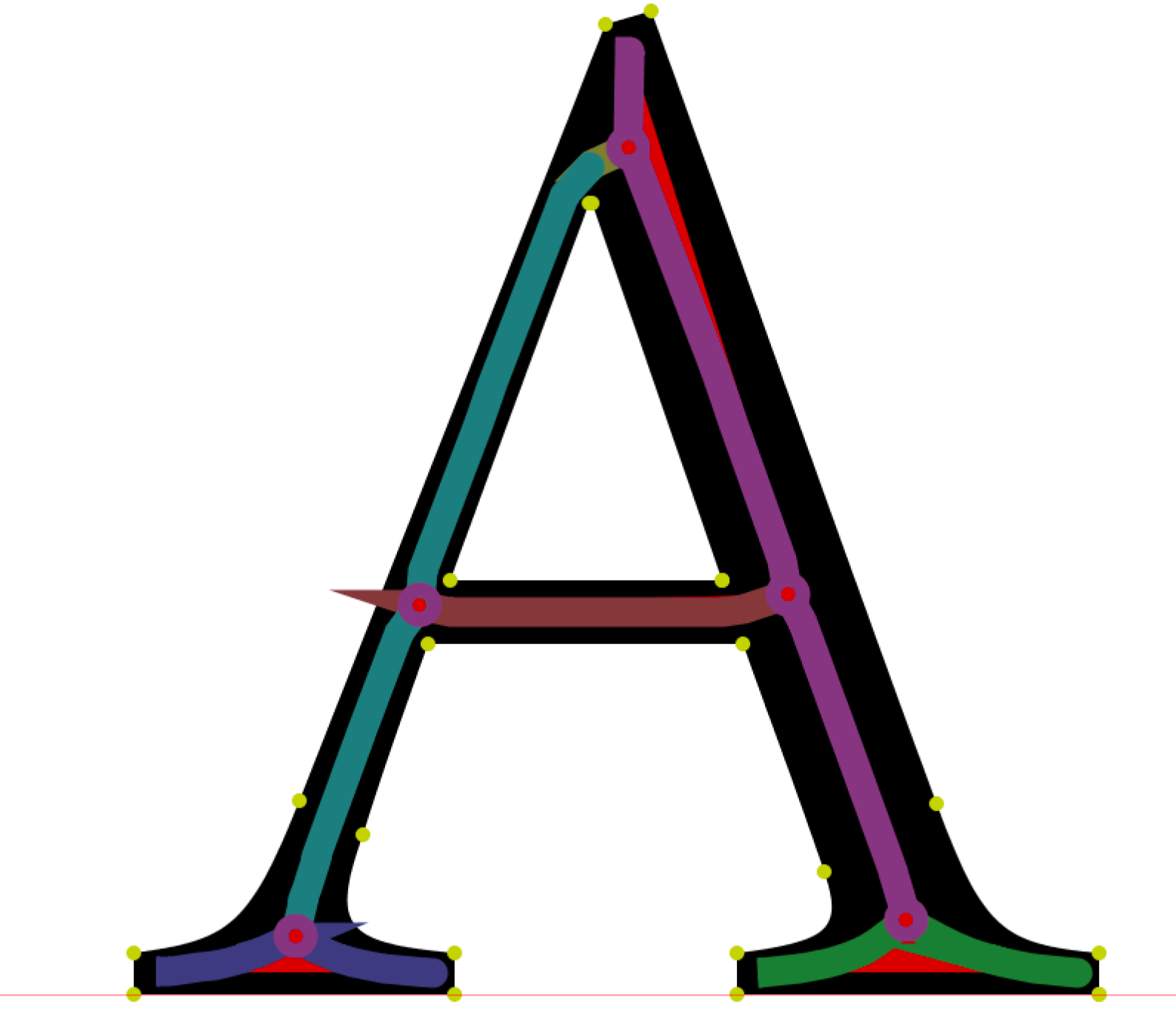
Nas imagens 101 a 103 pode ver-se como são selecionados e agrupados os traços do esqueleto que se intersectam num ponto.

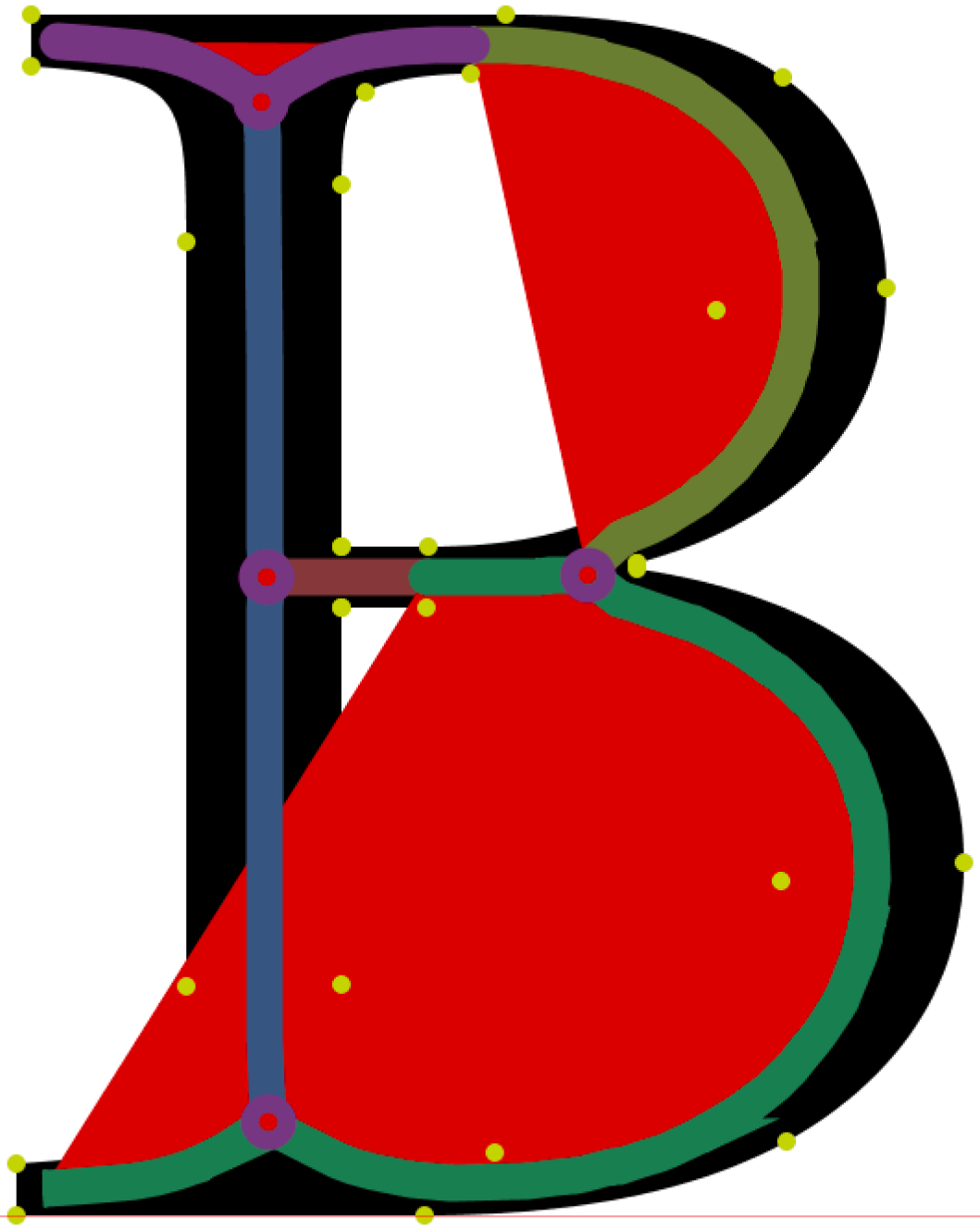


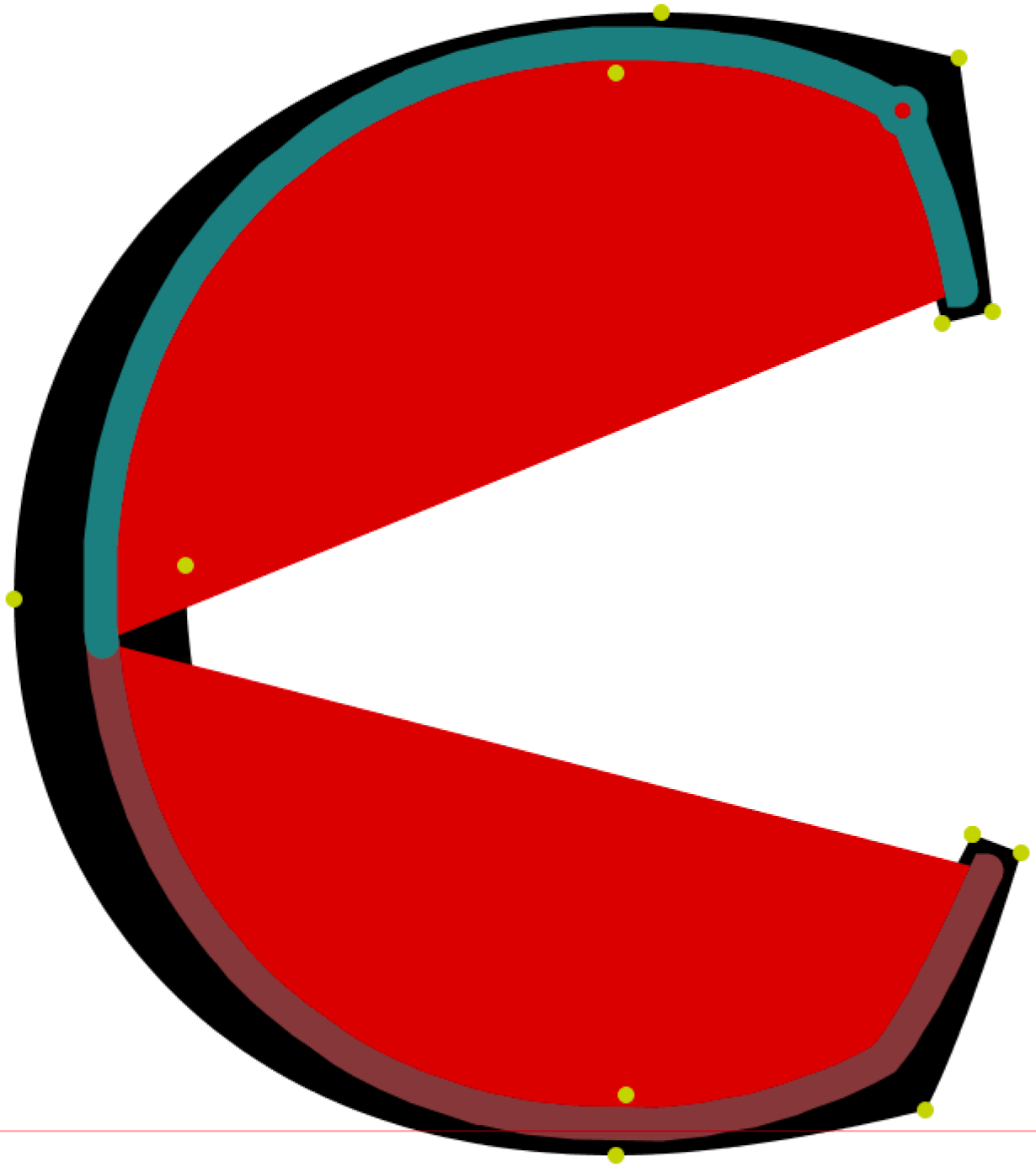
img 104
Janela do programa Glyph Surgeon. Na letra *k* os "ossos" do esqueleto foram agrupados resultando na identificação de 6 partes anatómicas, representadas pelas diferentes cores.

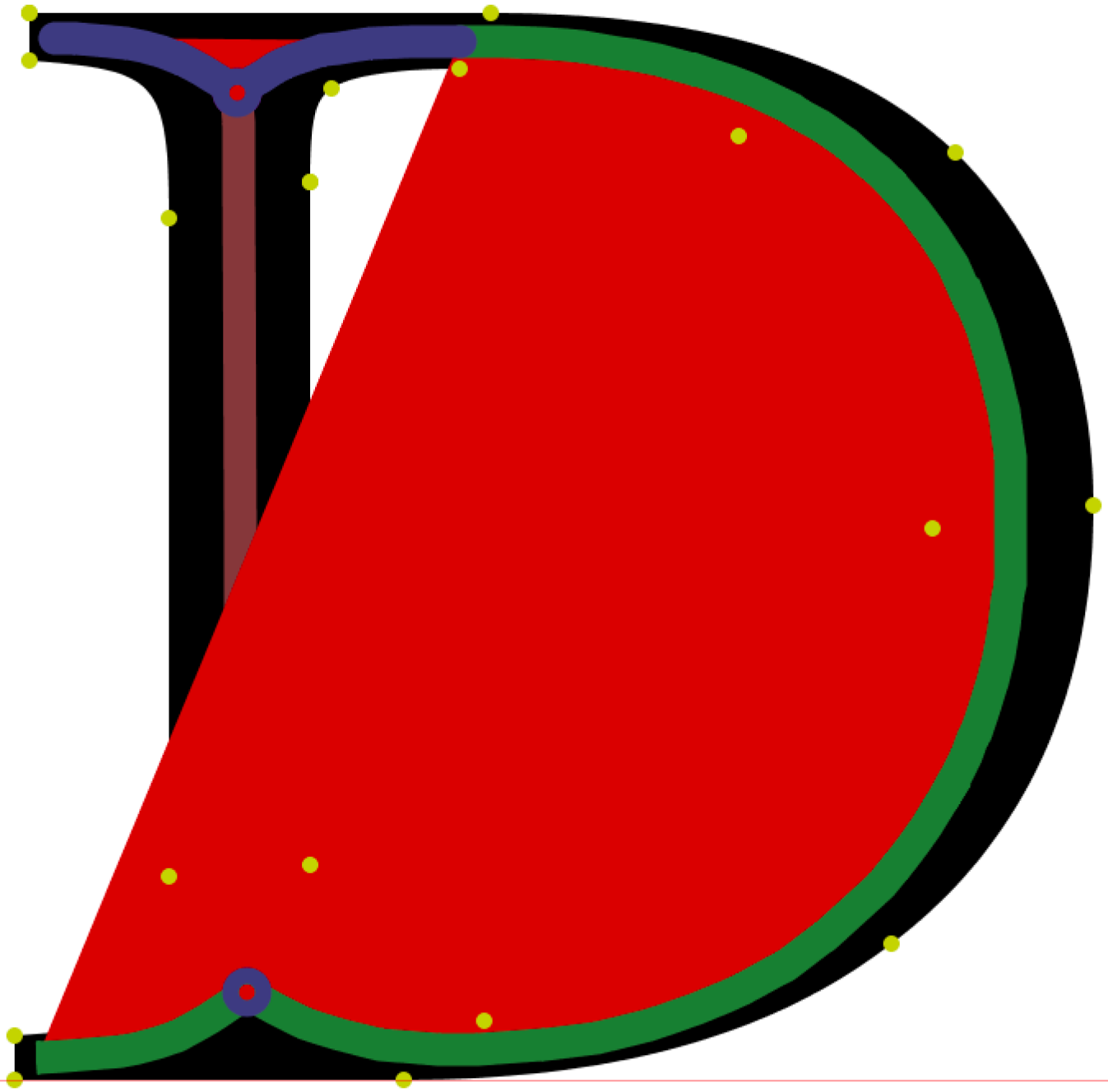
Posteriormente, para localizar as partes anatómicas, em cada "osso" de cada parte anatómica largura de cada traço à sua volta é analisada de forma a procurar os pontos onde o corte poderia ser feito.
Durante esta parte da implementação, a experimentação e o erro, levaram o processo a produzir resultados graficamente interessantes que consideramos pertinentes registar.

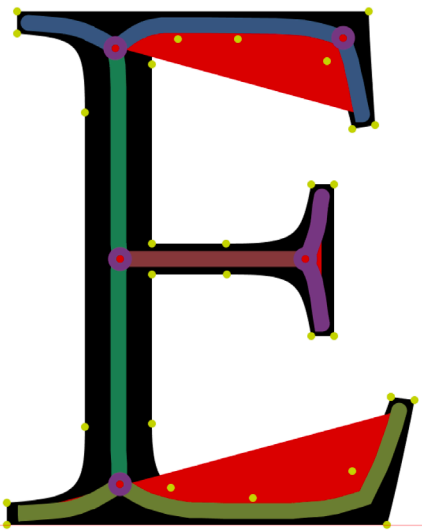
imgs 105 (pág. 92 a 113)
Erros gráficos

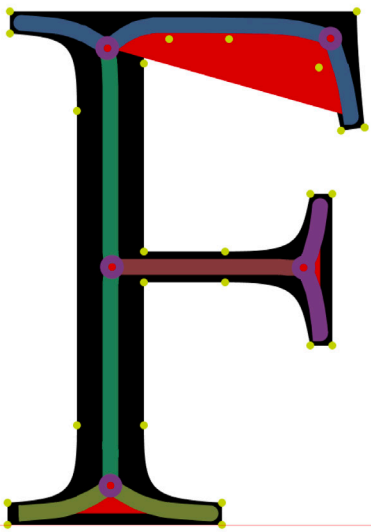


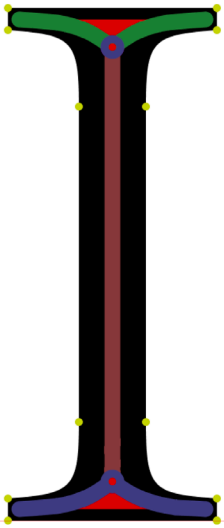


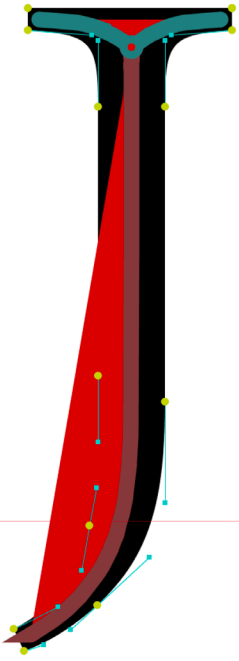


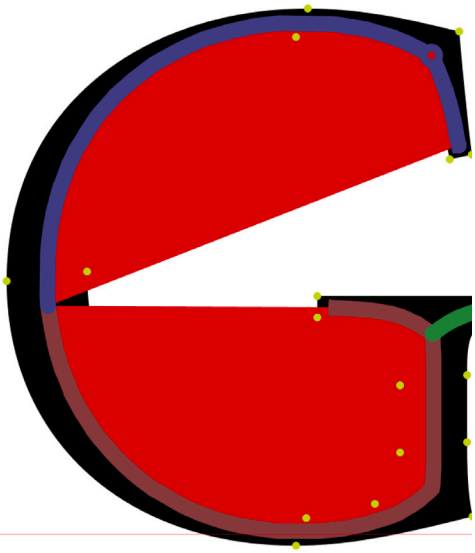


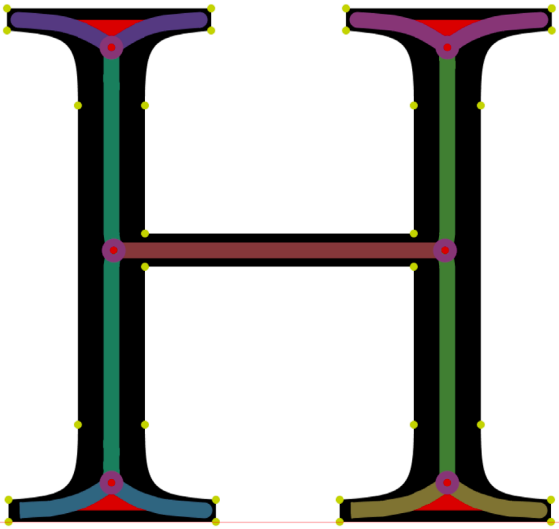


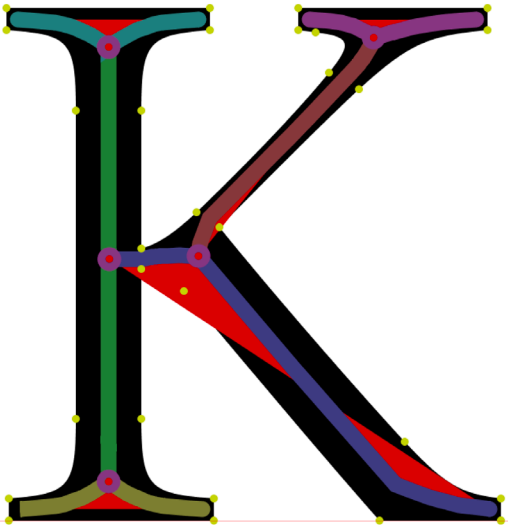


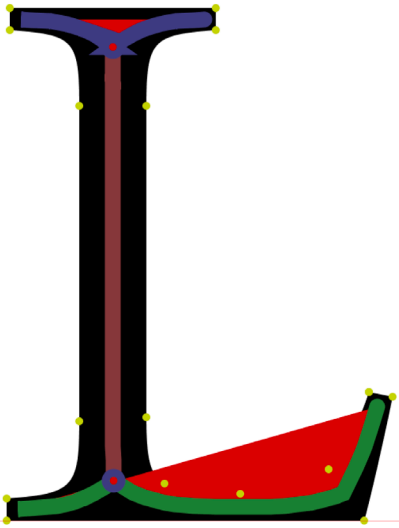


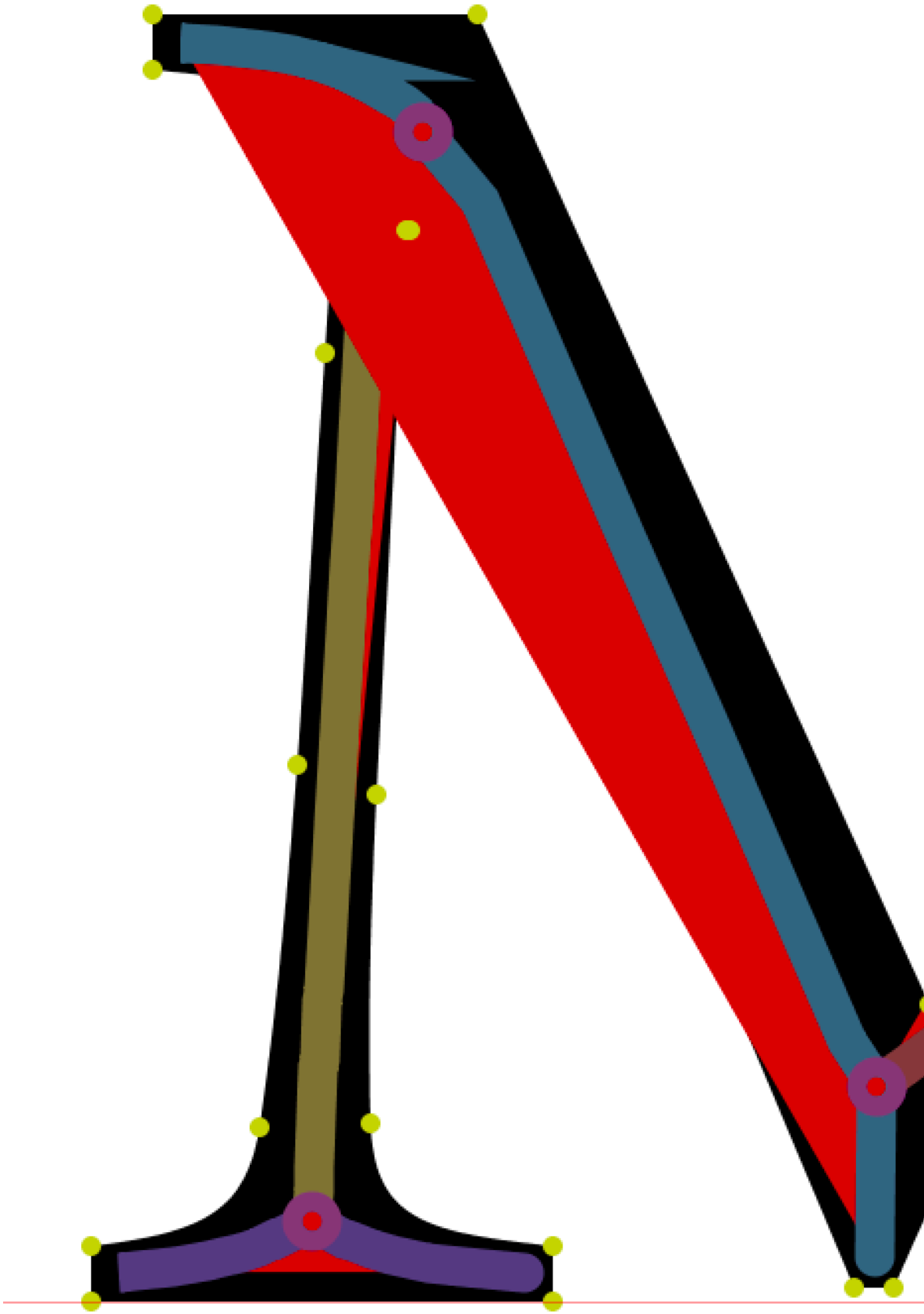


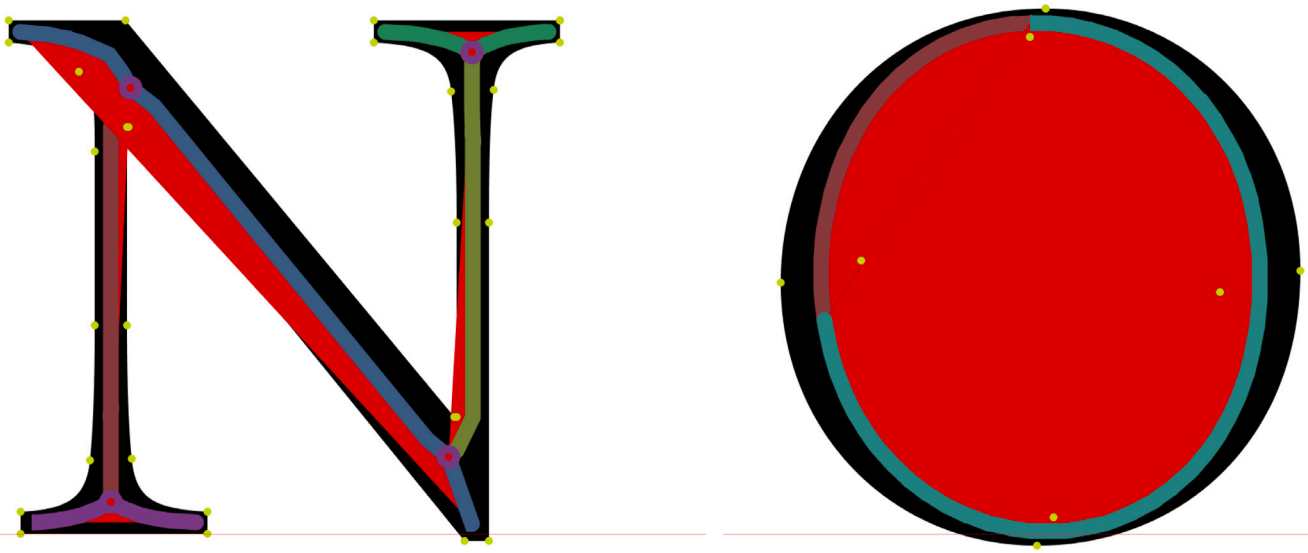


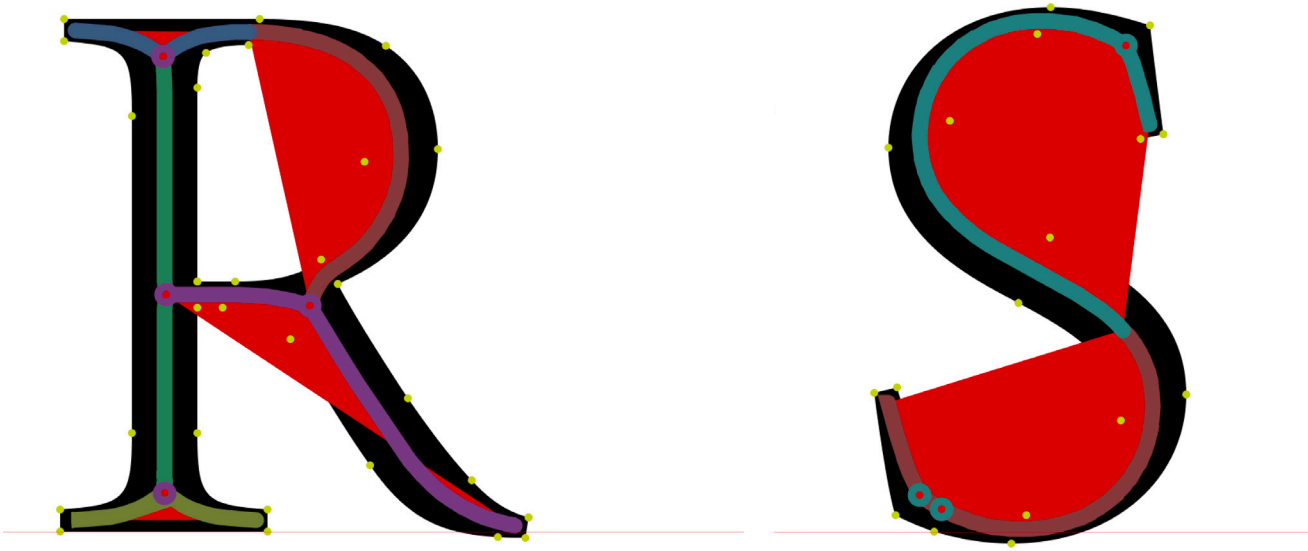


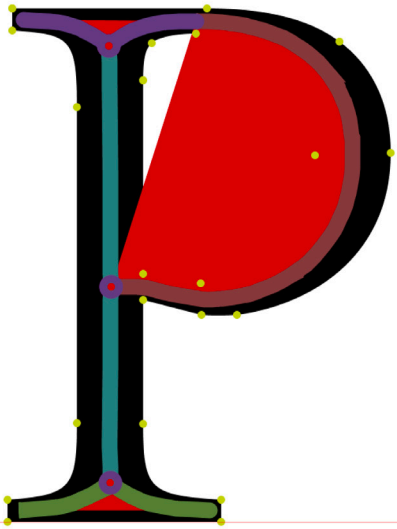


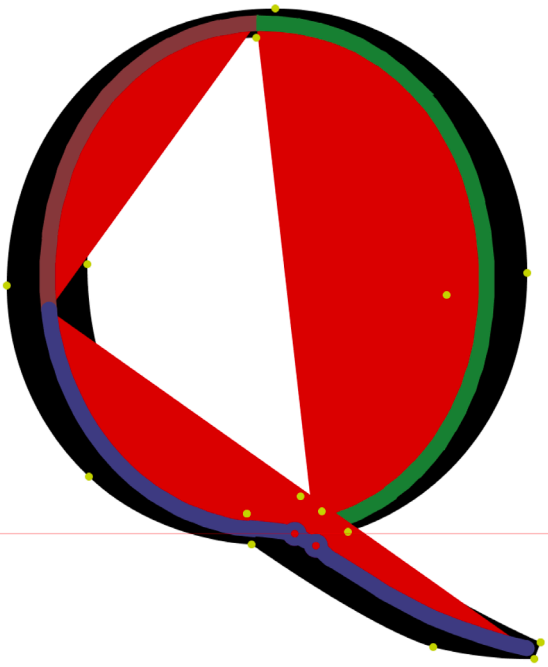


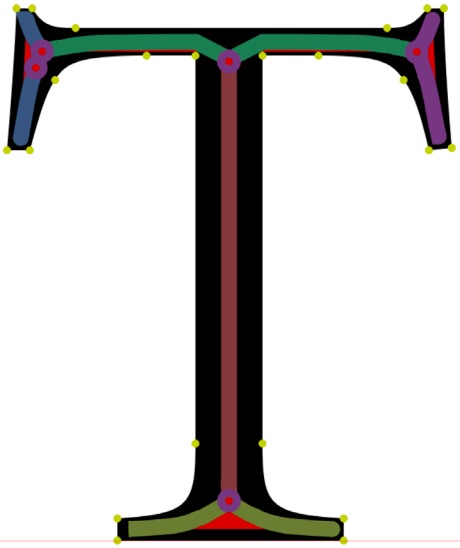


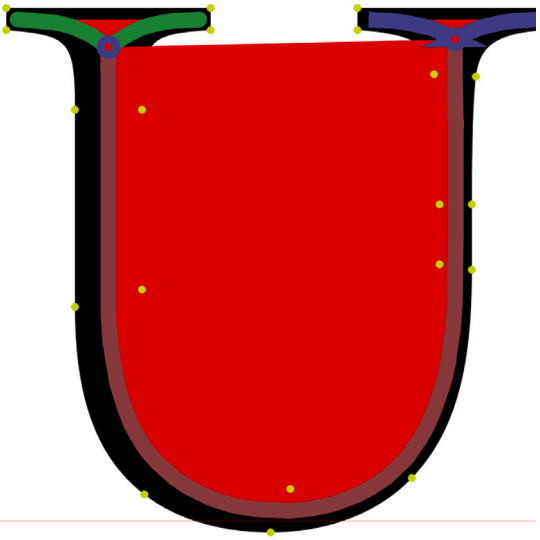


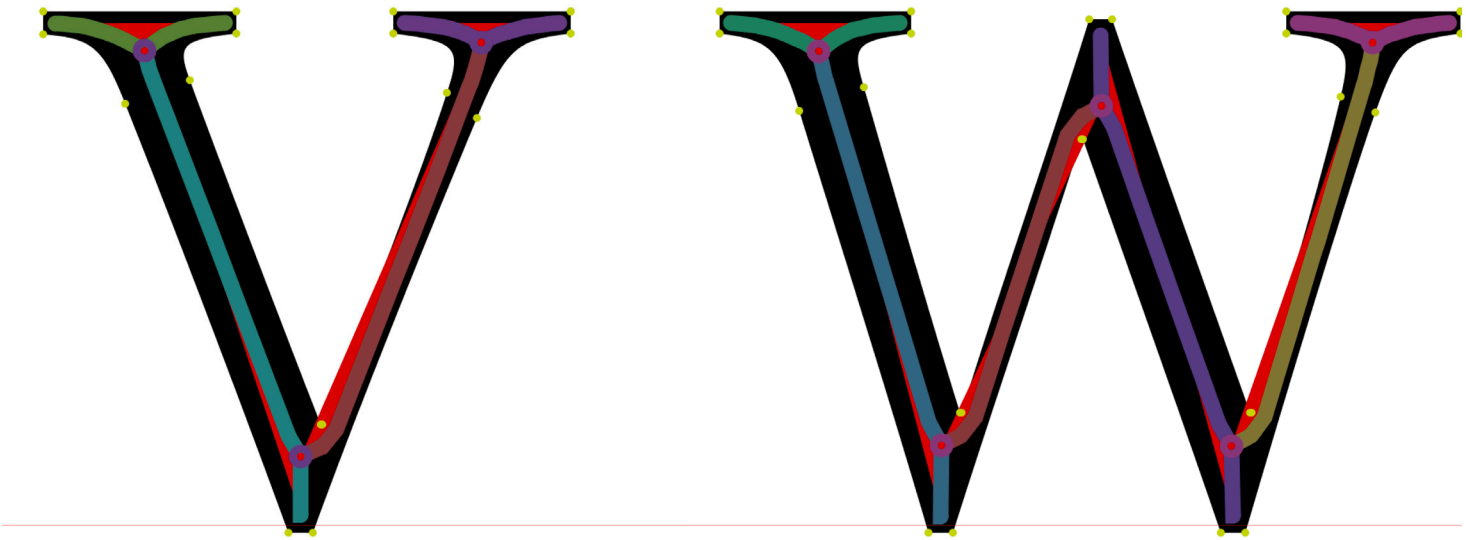


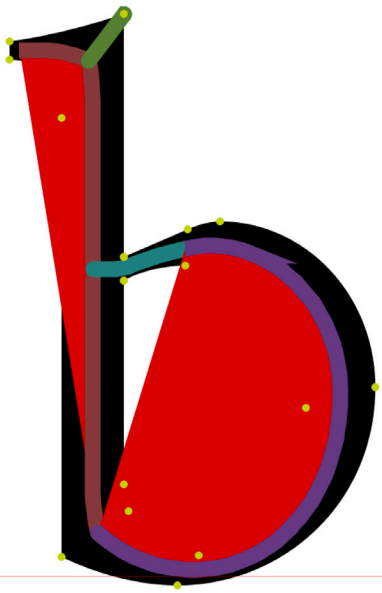


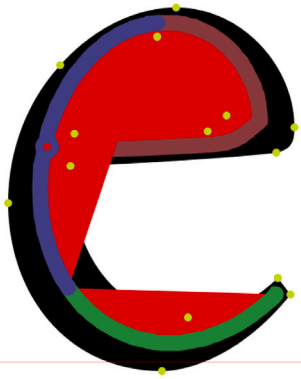


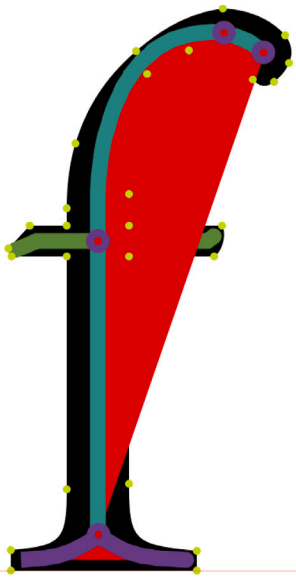


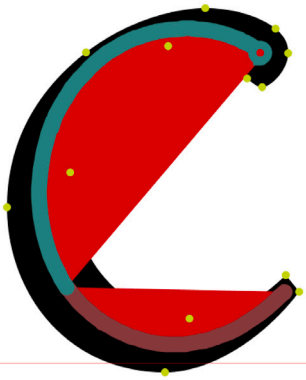


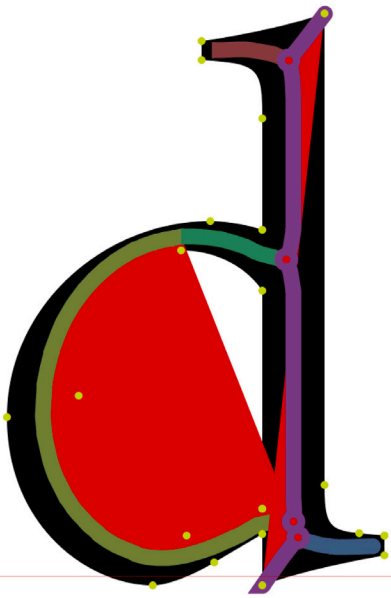


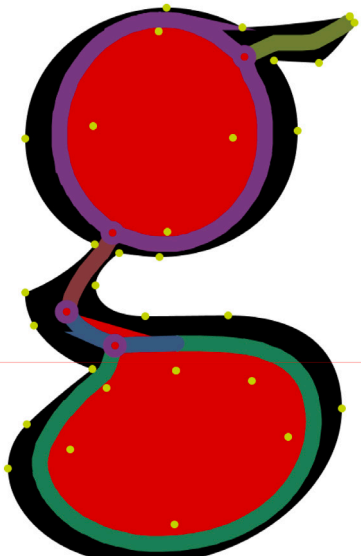


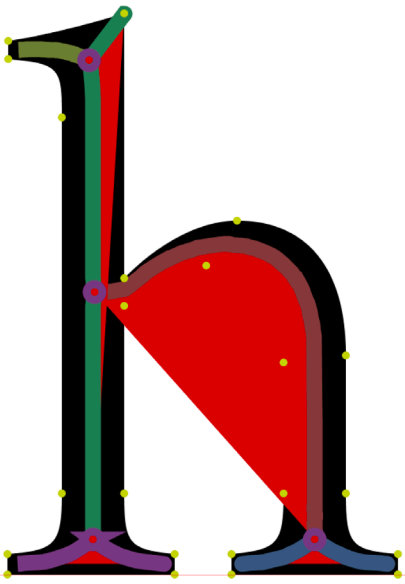


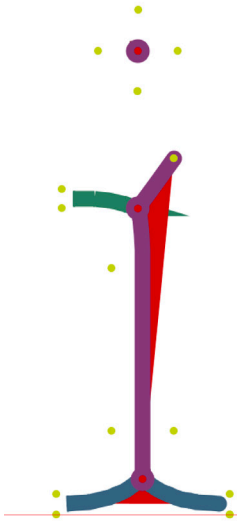


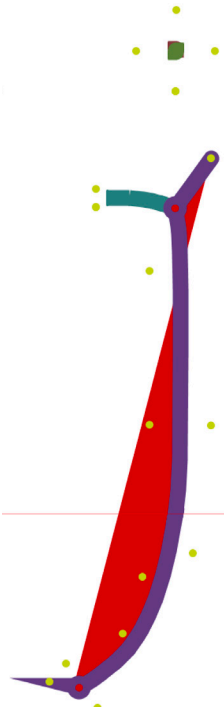


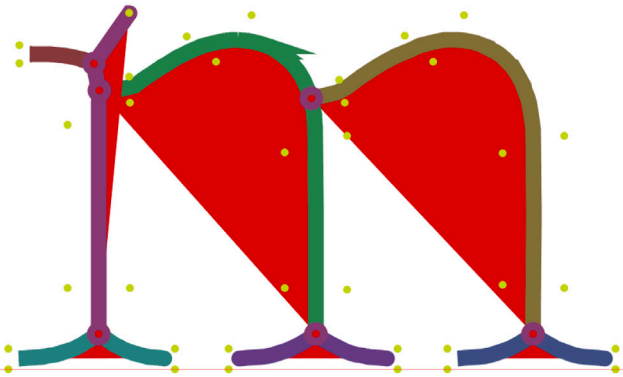


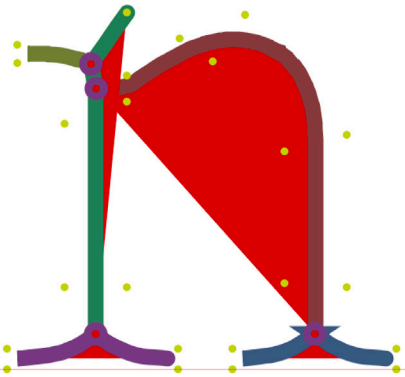


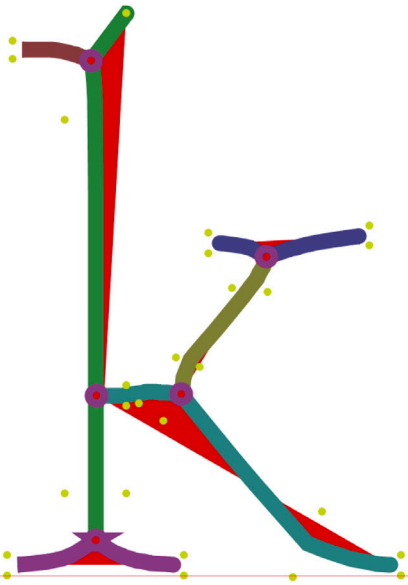


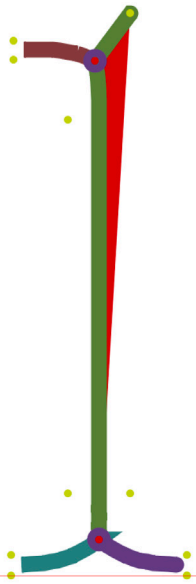


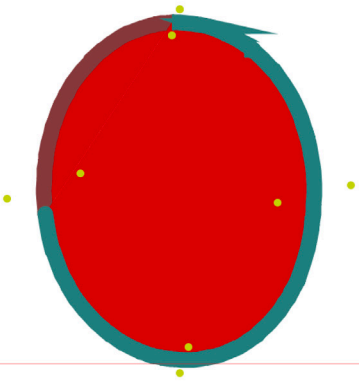


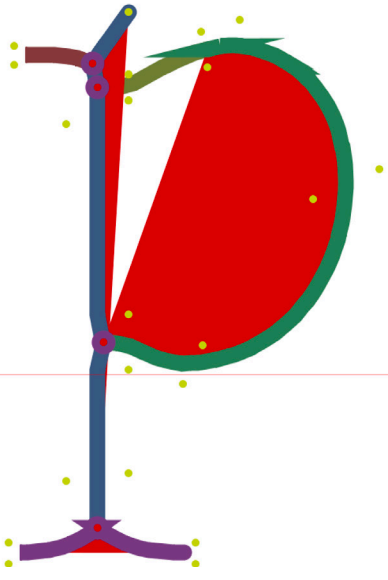


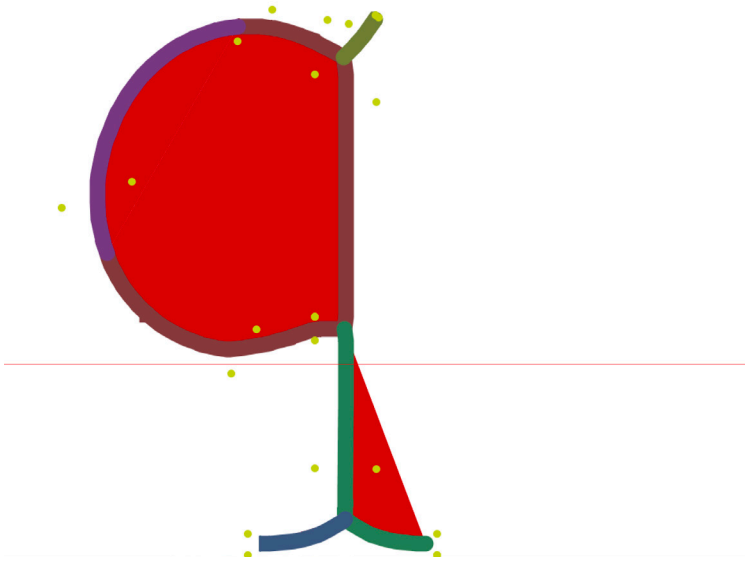


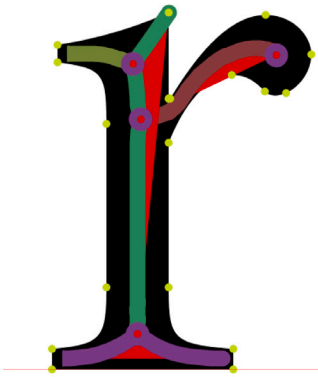


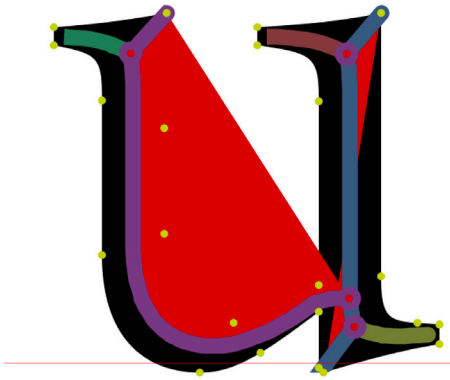


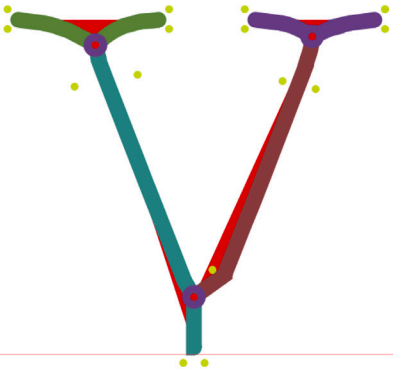


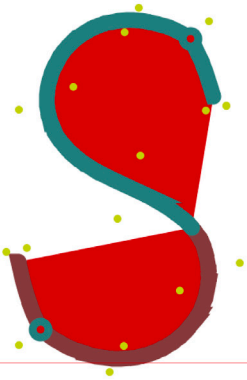


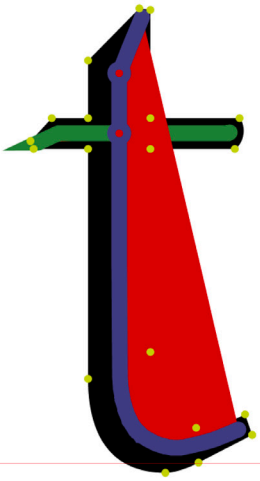


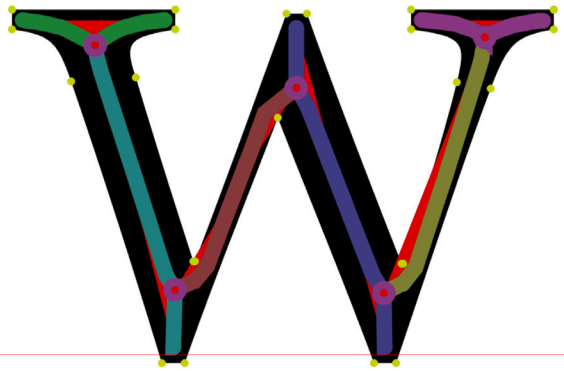


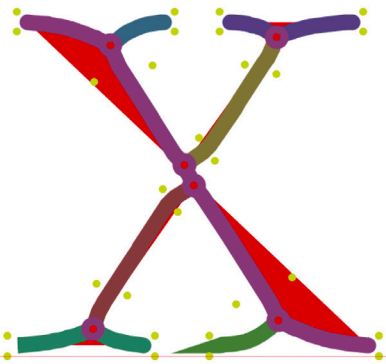


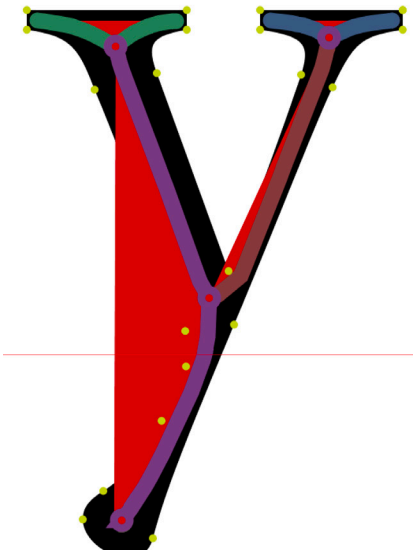


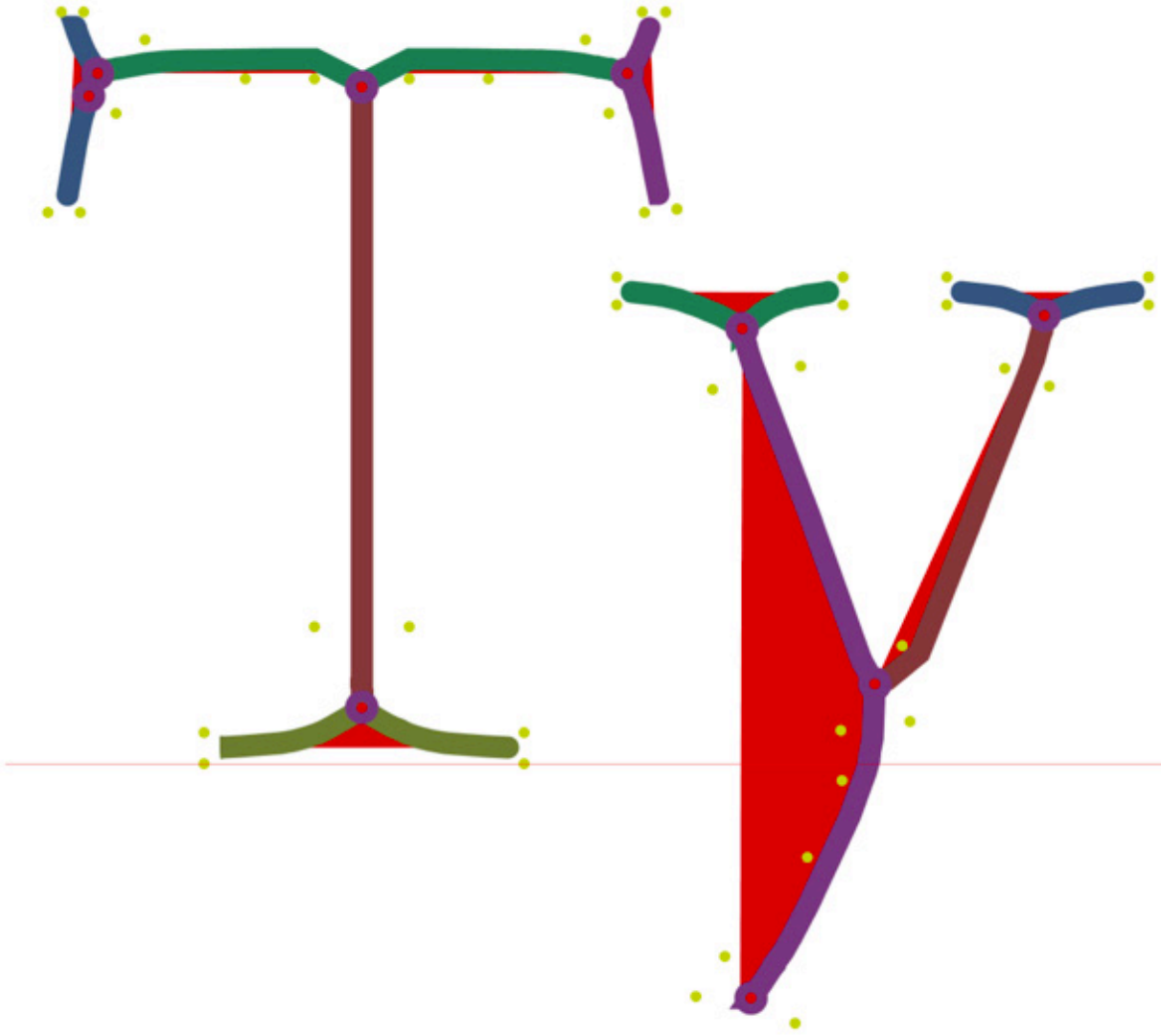


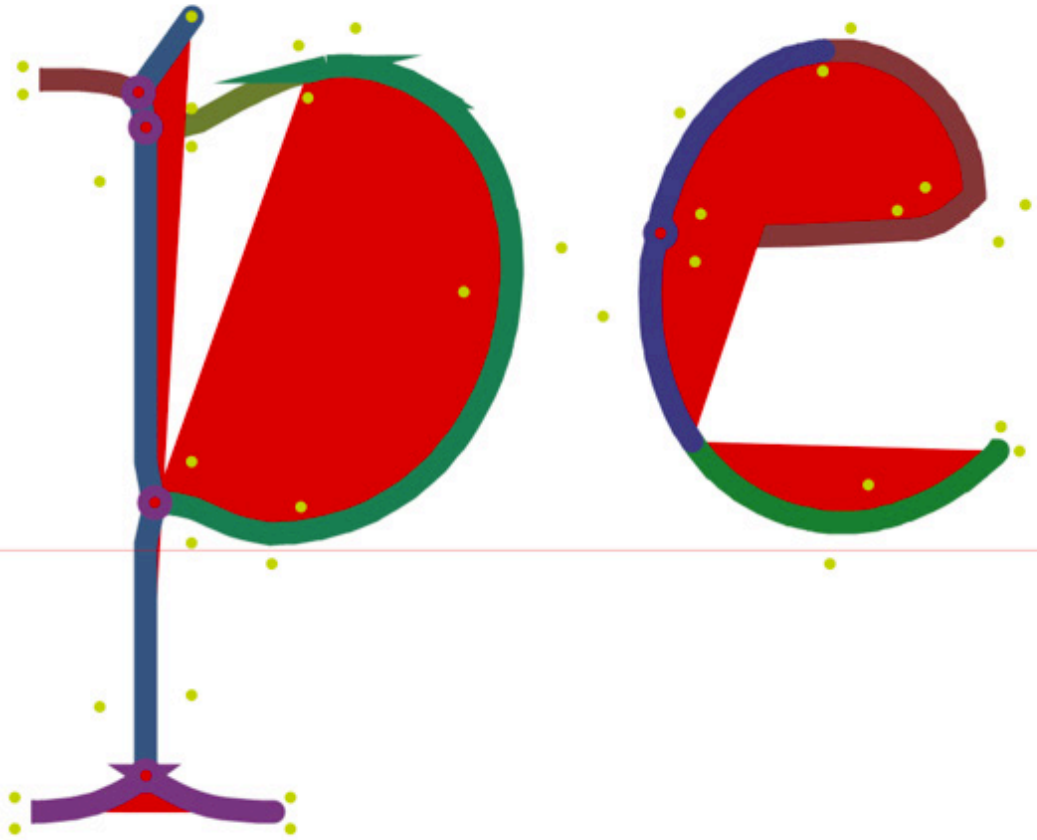








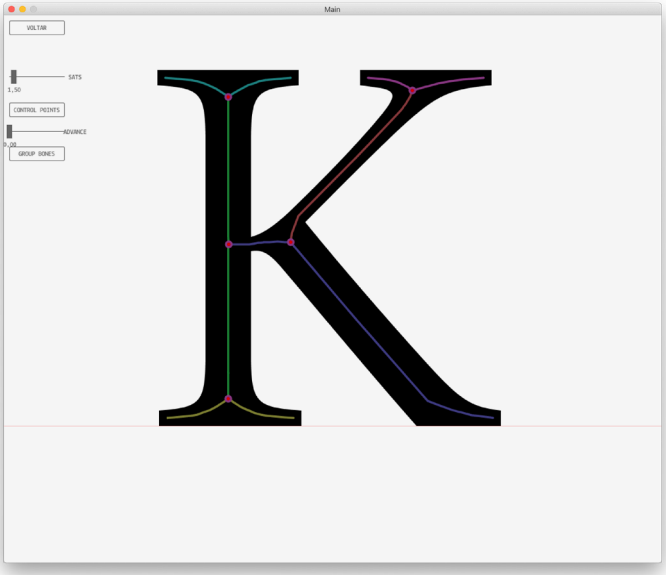




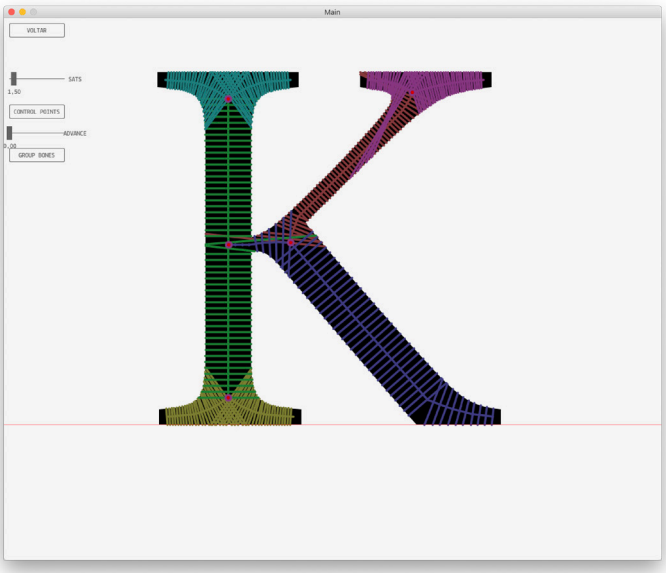
Tendo acesso aos "ossos" do esqueleto da letra e à sua separação e agrupação, é necessário detetar na forma qual a zona envolvente a cada um desses "ossos" de maneira a identificar as partes anatómicas efectivamente e as zonas onde o corte deverá acontecer.

Para isto, utilizaram-se novamente as funções desenvolvidas para deteção de intersecção entre linhas e curvas.

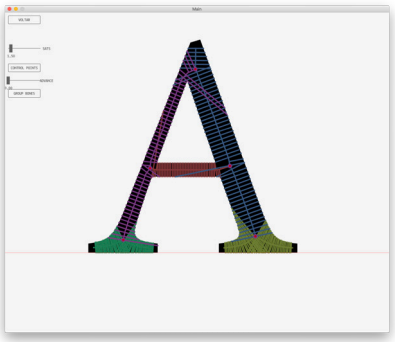
Para cada osso do esqueleto, a um intervalo constante, são projetadas linhas perpendiculares ao traço do esqueleto e a intersecção destas linhas com o contorno da forma são calculadas.

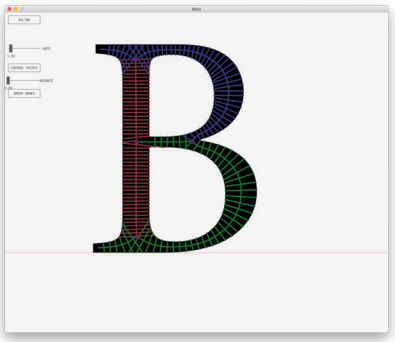


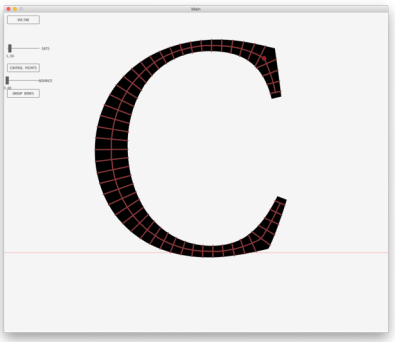
img 106 (topo)
Visualização dos ossos separados por parte anatómica da letra *K*.

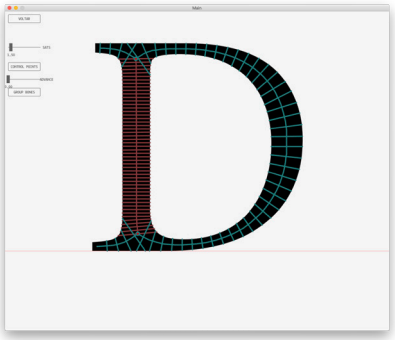


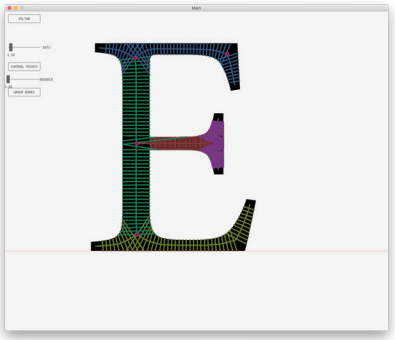
img 107 (baixo)
Aplicação do algoritmo descrito para identificação da zona envolvente de cada "osso".

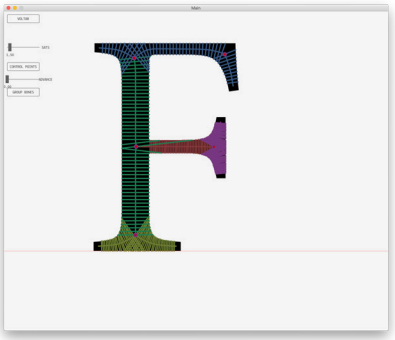

A

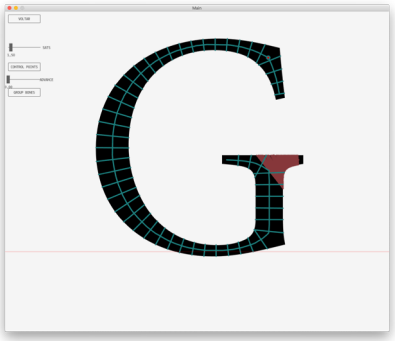

B

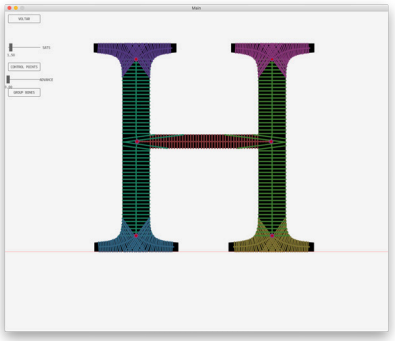

C

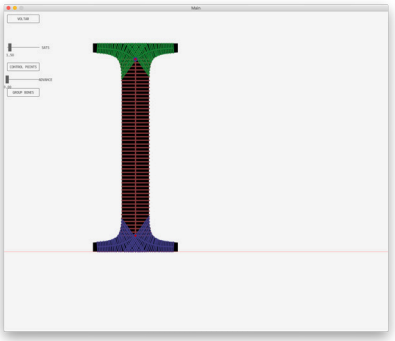

D

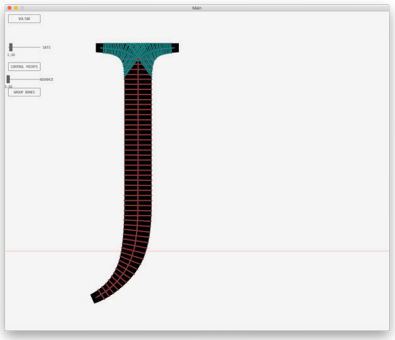

E

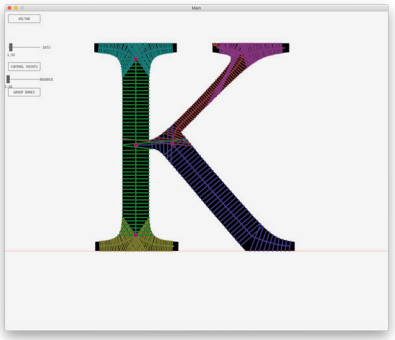

F

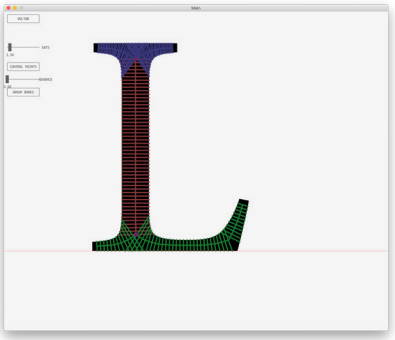

G

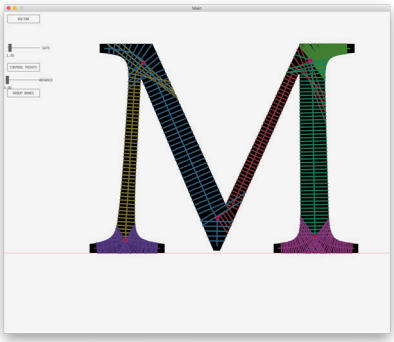

H

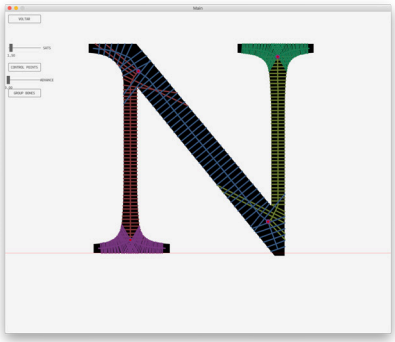

I

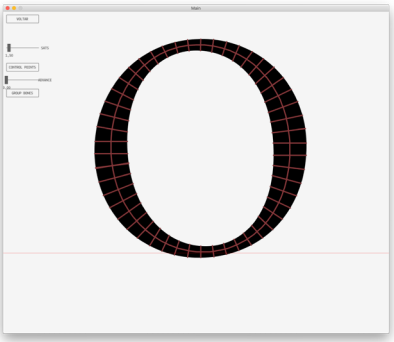

J

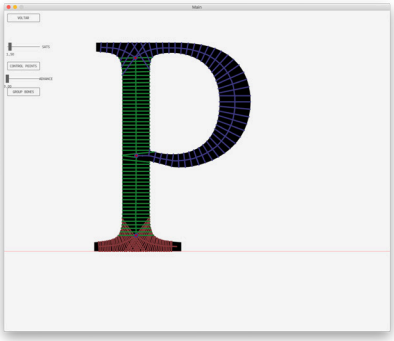

K

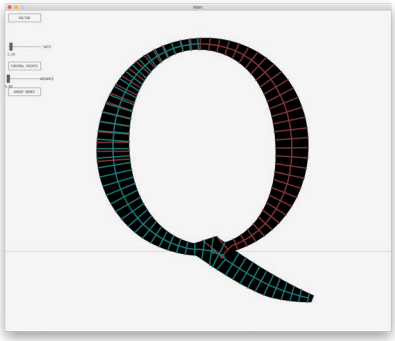

L

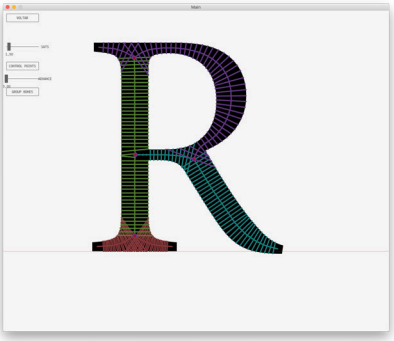

M

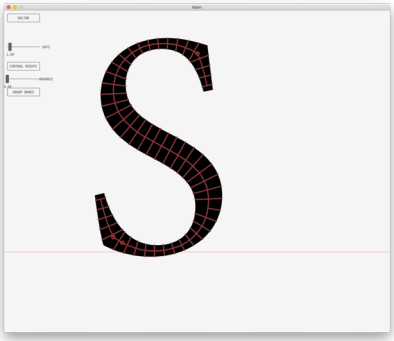

N

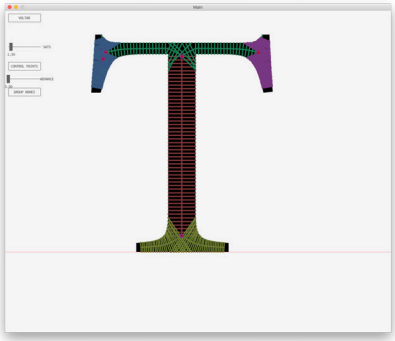

O

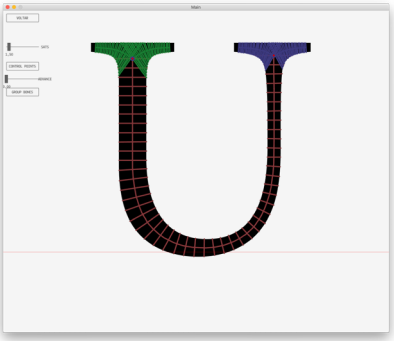

P

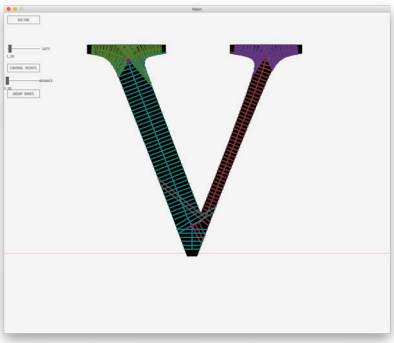

Q

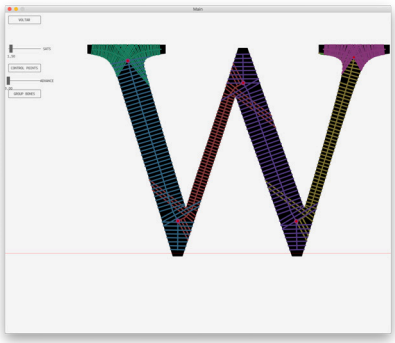

R

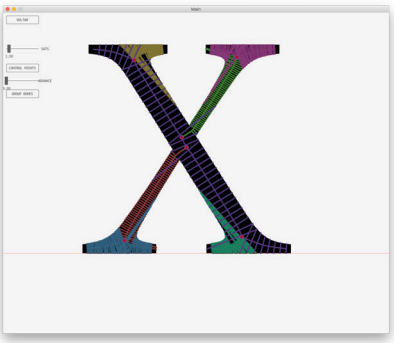

S

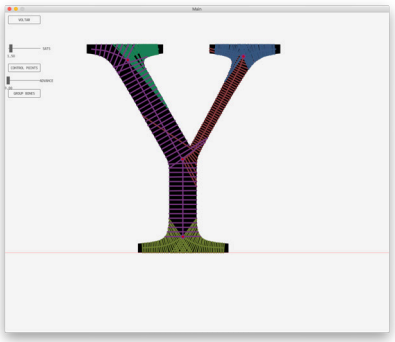

T

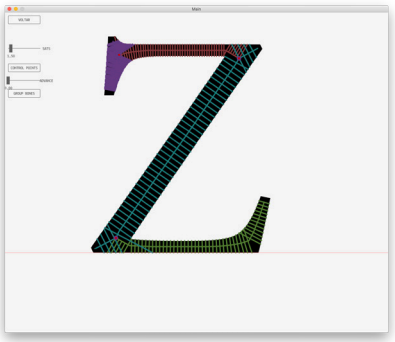

U

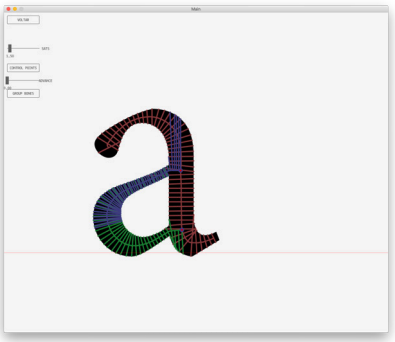

V

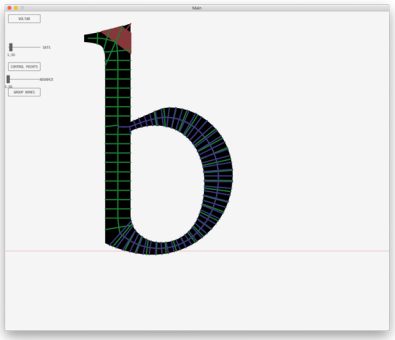

W

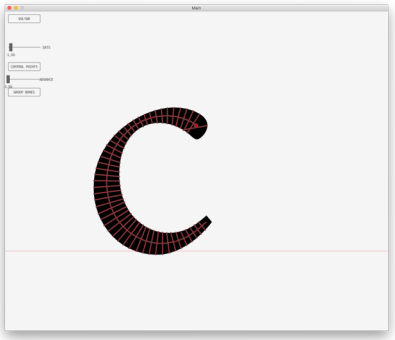

X

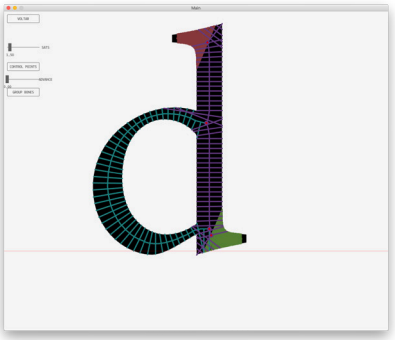


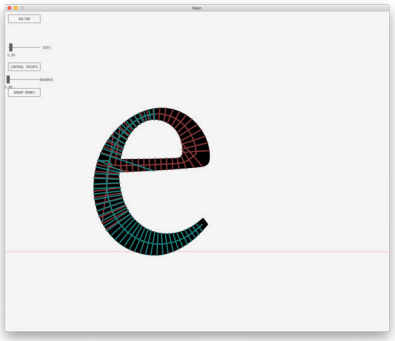


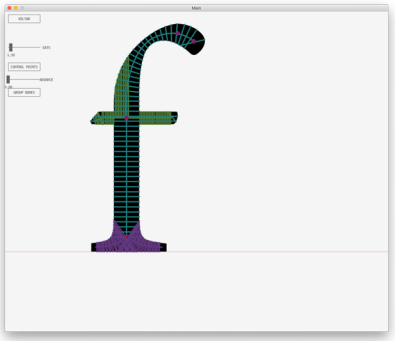


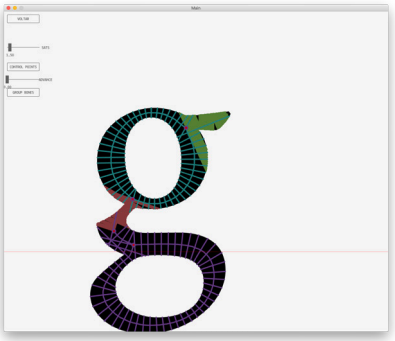


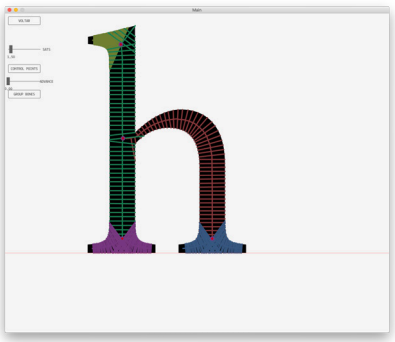


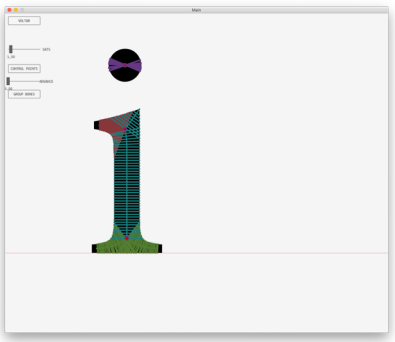


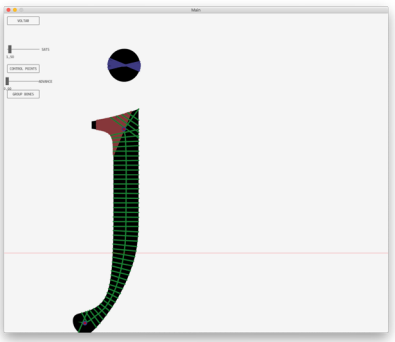


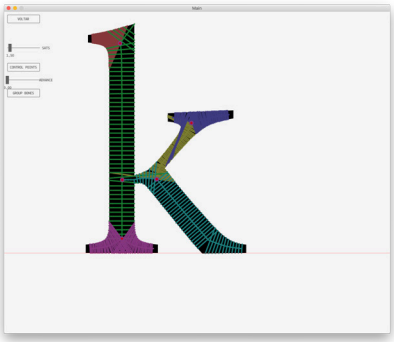
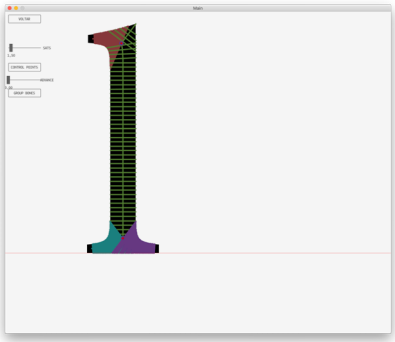
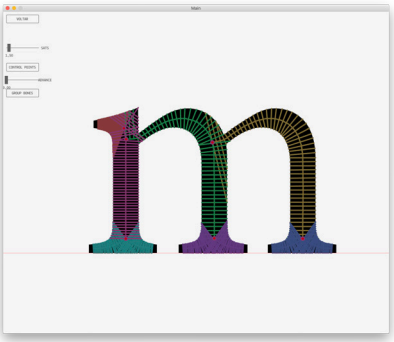
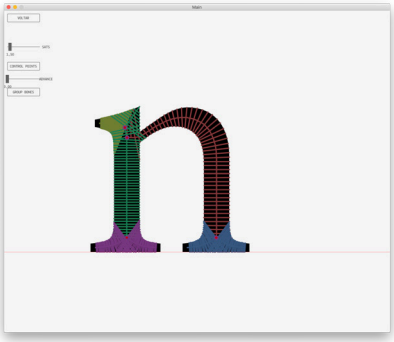
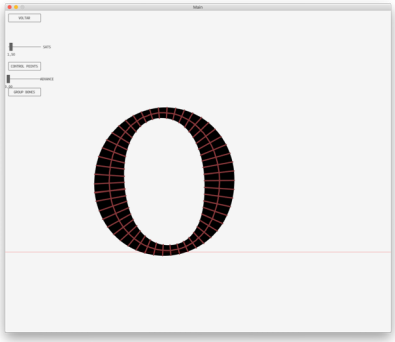
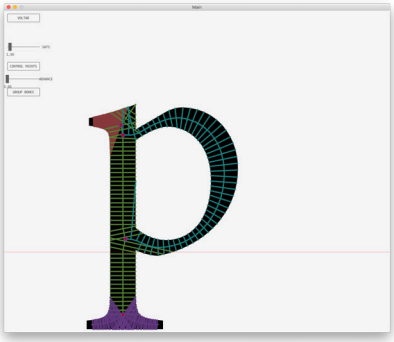
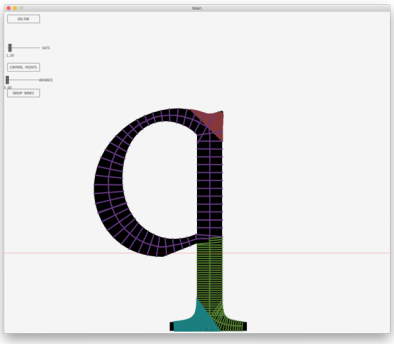
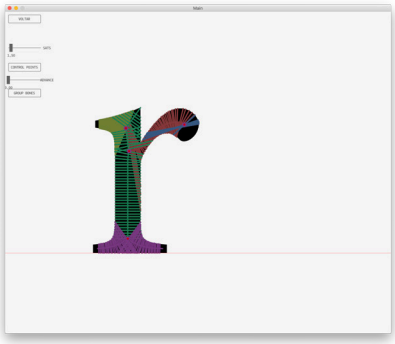
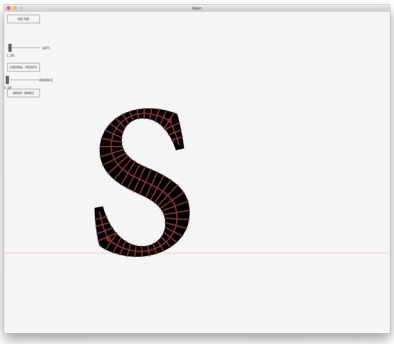
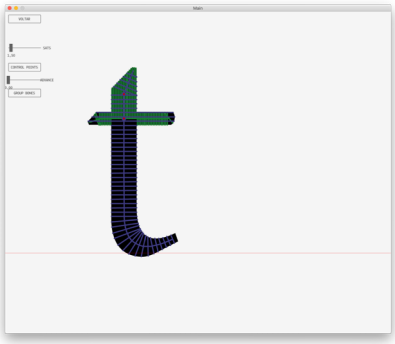
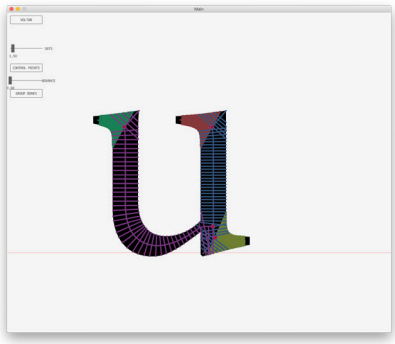
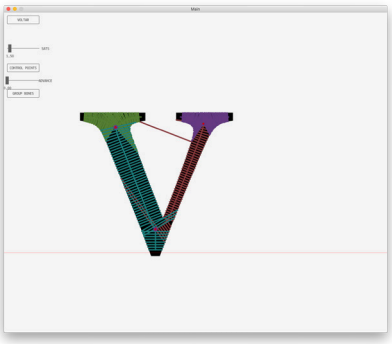


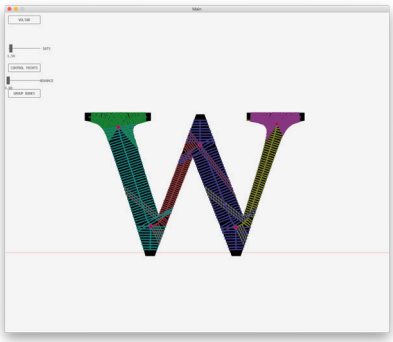


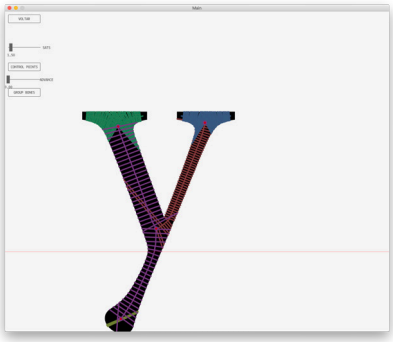


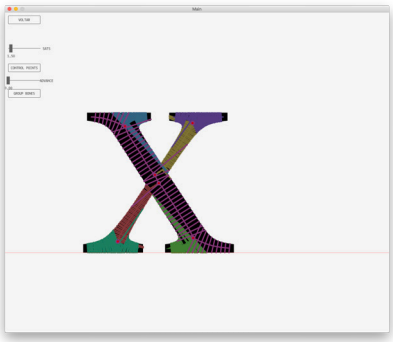


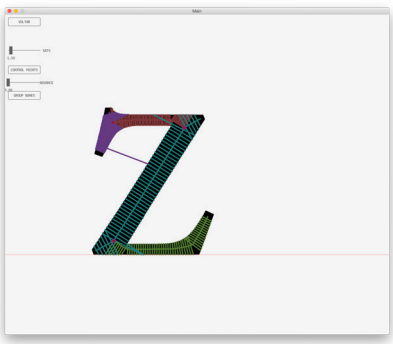










imgs 108 (pág. 115 a 119)
Janelas do programa Glyph Surgeon. Aplicação do método de localização da parte envolvente de cada "osso" a todas as maiúsculas e minúsculas do alfabeto.

Os resultados desta aplicação do método de deteção da zona envolvente mostraram-se promissores. No entanto, são identificáveis alguns artefactos inesperados em alguns glifos pontualmente.

Esta identificação primária das partes anatómicas é importante porque é através dessa identificação que prevemos que será possível identificar os pontos de corte da forma.

O tempo dedicado para o desenvolvimento já não permitiu que essa parte da implementação fosse realizada. No entanto uma ideia de como esses pontos poderiam ser identificados é a seguinte.

Criando uma forma através da uniam dos pontos de intersecção entre as perpendiculares o contorno da forma é possível identificar zonas de sobreposição. Essas zonas podem ser isoladas, e os vértices das formas criadas pela sobreposição podem ser utilizados para definir linhas de corte.

Este será à partida, o próximo e último passo a dar para conseguir obter a dissecação de glifos genericamente através das suas partes anatómicas.

Em termos de desenvolvimento, esta formulação de uma possível solução marca o fim do processo do trabalho prático realizado.

img 109 (topo esquerda, pág. oposta)
Identificação dos "ossos" do esqueleto e das zuas zonas envolventes.

img 110 (topo direita, pág. oposta)
Criação de um polígono com base nos pontos detectatos no passo anterior.

img 111 (baixo esquerda, pág. oposta)
Isolamento das sobreposições entre os polígonos criados.

img 112 (baixo direita, pág. oposta)
Seleção dos pontos de corte com base nos vértices das formas resultantes da sobreposição.

Conclusões e Trabalho Futuro

O processo de trabalho revelou, que as previsões feitas inicialmente estavam muito aquém do verdadeiro processo pelo qual seria preciso passar para a conclusão deste projeto. O objetivo final não foi alcançado, no entanto, reconhecemos o valor do trabalho realizado pelos avanços significativos para os quais contribui. Acredito que com este avanço, os resultados pretendidos não tardariam a aparecer, seria apenas necessário a existência de mais tempo disponível para o realizar do projeto.

Todo o processo foi extremamente enriquecedor, quer seja do ponto de vista teórico, no que toca à conceptualização do que é a letra e o valor que esta tem no nosso dia a dia. Não só como uma ferramenta de design, mas como um artefacto de design natural que muito revela acerca da nossa natureza humana. Quer seja do ponto de vista técnico e analítico. Um processo que foi muito estimulante e desafiador.

De qualquer forma, como já foi referido, a natureza do tema do projeto e a consciencialização do mesmo leva-me a concluir que imenso trabalho está por fazer. E sem dúvida que o desenvolver deste projeto foi um passo importante a nível pessoal para que melhor trabalho se venha a realizar no futuro.

Ideias para tarefas seguintes existem e estão conceptualizadas entre as quais:

- Separação das partes anatómicas recorrendo à sobreposição existente entre as manchas criadas pela análise da largura do glifo à volta de cada parte do esqueleto. Desta sobreposição resultam precisamente os pontos que poderiam ser utilizados.

- A parametrização do tipo de ligação entre as partes anatómicas poderá ser alterada através de algoritmos evolucionários de forma a que o resultado alcançado se aproxime de algum tipo de imagem.

- Documentação detalhada dos métodos de corte e dissecação de glifos.

- Desenvolvimento de uma biblioteca de geometria com as capacidades implementas otimizadas para utilização pública.

Bibliografia

(ref. 1) Pierre Jaquet-Droz - Biography, History and Inventions. (n.d.). Retrieved January 19, 2020, from https://history-computer.com/Dreamers/Jaquet-Droz.html

(ref. 2) Music Type - Music Printing History. (n.d.). Retrieved January 19, 2020, from https://www.musicprintinghistory.org/music-type

(ref. 3) Martin Majoor Type design. (n.d.). Retrieved January 19, 2020, from https://www.martinmajoor.com/1.1_scala_article_majoor.html

(ref. 4) interview with graphic designer david carson. (n.d.). Retrieved January 19, 2020, from https://www.designboom.com/design/interview-with-graphic-designer-david-carson-09-22-2013/

(ref. 5) Colophon Foundry. (n.d.). Retrieved January 19, 2020, from https://www.colophon-foundry.org/goods/pdu-specimen/

(ref. 6) Dries Wiewauters. (n.d.). Retrieved January 19, 2020, from http://www.drieswiewauters.eu/graphic/project_13/

(ref. 7) szaktöri — Fregio Mecano is a modular font of Italian origin... (n.d.). Retrieved January 19, 2020, from https://szaktori.tumblr.com/post/110538686080/fregio-mecano-is-a-modular-font-of-italian-origin/amp

(ref. 8) Grammato — Future is written. (n.d.). Retrieved January 19, 2020, from http://grammato.com/

(ref. 9) new visual identity by mevis en van deursen — Stedelijk Museum Amsterdam. (n.d.). Retrieved January 19, 2020, from https://www.stedelijk.nl/en/news/new-visual-identity-by-mevis-en-van-deursen

(ref. 10) Brand New: Stedelijk Museum Spells Out the Obvious. (n.d.). Retrieved January 19, 2020, from https://www.underconsideration.com/brandnew/archives/stedelijk_museum_amsterdam.php

(ref. 11) Catalog covers for Stedelijk Museum - Fonts In Use. (n.d.). Retrieved January 19, 2020, from https://fontsinuse.com/uses/3891/catalog-covers-for-stedelijk-museum

(ref. 12) Stedelijk Museum identity (2012) - Fonts In Use. (n.d.). Retrieved January 19, 2020, from https://fontsinuse.com/uses/12058/stedelijk-museum-identity-2012

(ref. 13) LUST | Graphic and Interactive Design | Grafisch en Interactief Ontwerp. (n.d.). Retrieved January 19, 2020, from https://lust.nl/#projects-5525

(ref. 14) LUST | Graphic and Interactive Design | Grafisch en Interactief Ontwerp. (n.d.). Retrieved January 19, 2020, from https://lust.nl/#projects-58

(ref. 15) The Affordances of Scripting Typography: The Manifesto. (n.d.). Retrieved January 19, 2020, from http://scriptingtypography.xyz/

(ref. 16) The Affordances of Scripting Typography | Diplom HGK 2018. (n.d.). Retrieved January 19, 2020, from https://nextgeneration.hgk.fhnw.ch/2018/diploma/visuelle-kommunikation/master/neves_pedro/2018-neves_pedro-the_affordances_of_scripting_typo/

Hatherly, A., & Portela, M. (1981). A Reinvenção da Leitura (Fac-Similada). d.

Sherman, W. H. (2014). Chapter 2. Toward a History of the Manicule. Used Books, (December 2004), 1–26. https://doi.org/10.9783/9780812203448.25

Meggs, P. B., Purvis, A. W., & Knipel, C. (2009). História do design gráfico. 720. Retrieved from http://books.google.com/books?id=o8ejQAAACAAJ&pgis=1

Sarnacki, K., & Saeed, K. (2018). Character Recognition Based on Skeleton Analysis. In L. J. Chmielewski, R. Kozera, A. Orłowski, K. Wojciechowski, A. M. Bruckstein, & N. Petkov (Eds.), Computer Vision and Graphics (pp. 148–159). Cham: Springer International Publishing.

Kamermans, M. (n.d.). A Primer on Bézier Curves. Retrieved November 18, 2020, from https://pomax.github.io/bezierinfo/#splitting

Amado, P., & Silva, C. (2011). Anatomia Tipográfica in Veloso, A.; Dias, N.; Martins, O.; Amado, P. "II Encontro de Tipografia: Livro de Atas". Aveiro: Edição eletrónica do II Encontro de Tipografia, Departamento de Comunicação e Arte da Universidade de Aveiro.

Saeed, K., Tabędzki, M., Rybnik, M., & Adamski, M. (n.d.). K3M: A universal algorithm for image skeletonization and a review of thinning techniques. International Journal of Applied Mathematics and Computer Science, 20(2), 317–335. https://doi.org/https://doi.org/10.2478/v10006-010-0024-4

Marxer, R. (n.d.). Geomerative. Retrieved November 18, 2020, from http://www.ricardmarxer.com/geomerative/

scikit-image: Image processing in Python — scikit-image. (n.d.). Retrieved November 18, 2020, from https://scikit-image.org/

Steenkamp, F. (n.d.). flo-mat - npm. Retrieved November 18, 2020, from https://www.npmjs.com/package/flo-mat

IMAGENS

1-5: Hatherly, A. (1975). A Reinvenção da Leitura.

6: de Rodes, S. (n.d.). Ovo. Retrieved November 18, 2020, from https://cdn.culturagenial.com/imagens/poema-o-ovo.jpg

7: Carmina Figurata. (n.d.). Retrieved November 18, 2020, from https://i.pinimg.com/564x/ea/5f/74/ea5f74b418bb1f64109a40eea8ae124a.jpg

8-9: Jaquet-Droz, P. (n.d.). The Writer. Retrieved January 29, 2020, from https://www.zmescience.com/other/feature-post/the-240-year-old-beautiful-ancestor-of-the-moderen-computer/

10: Movable Type. (n.d.). Retrieved November 18, 2020, from https://miro.medium.com/max/700/1*m296G9GngssSVOTxNiBBJg.jpeg

11: Music Type. (n.d.). Retrieved November 18, 2020, from https://musicprintinghistory.org/wp-content/uploads/2020/09/music-fonts-moveable-type.jpg

12: Martin Majoor. (n.d.). Scala Font. Retrieved November 18, 2020, from http://luc.devroye.org/fonts-26865.html

13: David Carson. (n.d.). Bryan Ferry Article. Retrieved November 18, 2020, from RayGun website: https://i.pinimg.com/originals/32/e1/c6/32e1c64b56bcc30b62563dd2baf340ac.jpg

15: Joseph A. David. (n.d.). PDU. Retrieved November 18, 2020, from https://dspncdn.com/a1/media/692x/5e/a1/50/5ea15020496161ddf167fdab0a1d76b2.jpg

16: Fregio Mecano. (n.d.). Retrieved November 18, 2020, from https://i.pinimg.com/originals/b1/e4/cf/b1e4cf858b32e1e86860fc28700d38b5.jpg

17: Grammato — Future is written. (n.d.). Retrieved November 18, 2020, from https://www.grammato.com/

18: Mevis & Van Deursen. (n.d.). Stedelijk Museum Amsterdam Logo. Retrieved November 18, 2020, from https://artmo.com/wp-content/uploads/ultimatemember/11728/profile_photo.png?1578039063

19: Mevis & Van Deursen. (n.d.). Stedelijk Museum Wayfinding. Retrieved November 18, 2020, from https://i.pinimg.com/originals/03/43/f0/0343f08aa9a9d132d52495c661afcc9e.png

20: Lust. (n.d.). Type Dynamics. Retrieved November 18, 2020, from https://lust.nl/#projects-5525

21: Lust. (n.d.). Scrapper Type. Retrieved November 18, 2020, from https://lust.nl/#projects-58

22: Pedro Neves. (n.d.). The Affordances of Scripting Typography: The Manifesto. Retrieved November 18, 2020, from http://scriptingtypography.xyz/

73: getShape() - Type of vertex was wrong for Cubic by hugovalepereira · Pull Request #6092 · processing/processing. (n.d.). Retrieved November 18, 2020, from https://github.com/processing/processing/pull/6092

As restantes imagens são da minha autoria.