Mele Sotiriou projet bezier

Generated by Doxygen 1.12.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

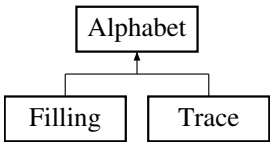
Class Documentation

4.1 Alphabet Class Reference

Parent class for Trace and Filling.

`#include <Alphabet.h>`

Inheritance diagram for Alphabet:



Public Member Functions

- **Alphabet** ()=default

  Default Constructor for class Alphabet.
- virtual void generateAlphabet (int x)=0

  Produces all the points for the traced version of the alphabet.
- virtual ~**Alphabet** ()

  Default Destructor for class Alphabet.

Public Attributes

- vector< vector< vector< double > > > **letters**

Protected Attributes

- vector< vector< double > > **results**

4.1.1 Detailed Description

Parent class for Trace and Filling.

4.1.2 Member Function Documentation

4.1.2.1 generateAlphabet()

```
virtual void Alphabet::generateAlphabet (
            int x)  [pure virtual]
```

Produces all the points for the traced version of the alphabet.

Parameters

x	Distinguishes between the two alphabets that need to be traced/filled

Implemented in Filling, and Trace.

The documentation for this class was generated from the following file:

- Alphabet.h

4.2 DeCasteljau Class Reference

Implements the DeCasteljau algorithm.

`#include <DeCasteljau.h>`

Static Public Member Functions

- static vector< vector< double > > linearBezier (vector< double > p0, vector< double > p1)

  Creates a straight line between p0 and p1.
- static vector< vector< double > > quadraticBezier (vector< double > p0, vector< double > p1, vector< double > p2)

  Creates a curve between p0 and p2 with p1 determining the curvature.
- static vector< vector< double > > cubicBezier (vector< double > p0, vector< double > p1, vector< double > p2, vector< double > p3)

  Creates a curve between p0 and p3 with p1 and p2 determining the curvature.

Static Protected Member Functions

- static vector< double > interpolation (vector< double > p0, vector< double > p1, double t)

  Returns the interpolated point between p0 and p1 for time t.

4.2.1 Detailed Description

Implements the DeCasteljau algorithm.

4.2.2 Member Function Documentation

4.2.2.1 cubicBezier()

```
vector< vector< double > > DeCasteljau::cubicBezier (
            vector< double > p0,
            vector< double > p1,
            vector< double > p2,
            vector< double > p3)  [static]
```

Creates a curve between p0 and p3 with p1 and p2 determining the curvature.

Parameters

p0	the first point
p1	the second point
p2	the third point
p3	the fourth point

Returns

a vector containing all the points of the curve to be drawn

4.2.2.2 interpolation()

```
vector< double > DeCasteljau::interpolation (
            vector< double > p0,
            vector< double > p1,
            double t)  [static], [protected]
```

Returns the interpolated point between p0 and p1 for time t.

Parameters

p0	the first point
p1	the second point
t	the time parameter of the interpolation

Returns

a vector containing 2 doubles (the x and y coordinates of a point)

4.2.2.3 linearBezier()

```
vector< vector< double > > DeCasteljau::linearBezier (
            vector< double > p0,
            vector< double > p1)  [static]
```

Creates a straight line between p0 and p1.

Parameters

p0	the first point
p1	the second point

Returns

a vector containing all the points of the line to be drawn

4.2.2.4 quadraticBezier()

```
vector< vector< double > > DeCasteljau::quadraticBezier (
            vector< double > p0,
            vector< double > p1,
            vector< double > p2)  [static]
```

Creates a curve between p0 and p2 with p1 determining the curvature.

Parameters

p0	the first point
p1	the second point
p2	the third point

Returns

a vector containing all the points of the curve to be drawn

The documentation for this class was generated from the following files:

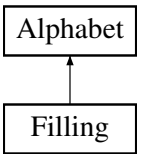
- DeCasteljau.h
- DeCasteljau.cpp

4.3 Filling Class Reference

Creates an alphabet with every letter filled.

`#include <Filling.h>`

Inheritance diagram for Filling:



Public Member Functions

- void generateAlphabet (int x) override

  Produces every point needed for a filled font.
- vector< vector< double > > createA (double offsetY)

  Creates a filled-in version of A.
- vector< vector< double > > createB (double offsetX, double offsetY)

  Creates a filled-in version of C.
- vector< vector< double > > createC (double offsetX, double offsetY)

  Creates a filled-in version of C.
- vector< vector< double > > createD (double offsetX, double offsetY)

  Creates a filled-in version of D.
- vector< vector< double > > createE (double offsetX, double offsetY)

  Creates a filled-in version of E.
- vector< vector< double > > createF (double offsetX, double offsetY)

  Creates a filled-in version of F.
- vector< vector< double > > createG (double offsetX, double offsetY)

  Creates a filled-in version of G.
- vector< vector< double > > createH (double offsetX, double offsetY)

Creates a filled-in version of H.

- vector< vector< double > > createI (double offsetX, double offsetY)

  Creates a filled-in version of I.
- vector< vector< double > > createJ (double offsetX, double offsetY)

  Creates a filled-in version of J.
- vector< vector< double > > createK (double offsetX, double offsetY)

  Creates a filled-in version of K.
- vector< vector< double > > createL (double offsetX, double offsetY)

  Creates a filled-in version of L.
- vector< vector< double > > createM (double offsetX, double offsetY)

  Creates a filled-in version of M.
- vector< vector< double > > createN (double offsetY)

  Creates a filled-in version of N.
- vector< vector< double > > createO (double offsetX, double offsetY)

  Creates a filled-in version of O.
- vector< vector< double > > createP (double offsetX, double offsetY)

  Creates a filled-in version of P.
- vector< vector< double > > createQ (double offsetX, double offsetY)

  Creates a filled-in version of Q.
- vector< vector< double > > createR (double offsetX, double offsetY)

  Creates a filled-in version of R.
- vector< vector< double > > createS (double offsetX, double offsetY)

  Creates a filled-in version of S.
- vector< vector< double > > createT (double offsetX, double offsetY)

  Creates a filled-in version of T.
- vector< vector< double > > createU (double offsetX, double offsetY)

  Creates a filled-in version of U.
- vector< vector< double > > createV (double offsetX, double offsetY)

  Creates a filled-in version of V.
- vector< vector< double > > createW (double offsetX, double offsetY)

  Creates a filled-in version of W.
- vector< vector< double > > createX (double offsetX, double offsetY)

  Creates a filled-in version of X.
- vector< vector< double > > createY (double offsetX, double offsetY)

  Creates a filled-in version of Y.
- vector< vector< double > > createZ (double offsetX, double offsetY)

  Creates a filled-in version of Z.
- ~**Filling** () override=default

  Standard destructor for class Filling.

Public Member Functions inherited from Alphabet

- **Alphabet** ()=default

  Default Constructor for class Alphabet.
- virtual ~**Alphabet** ()

  Default Destructor for class Alphabet.

Additional Inherited Members

Public Attributes inherited from Alphabet

- vector< vector< vector< double > > > **letters**

Protected Attributes inherited from Alphabet

- vector< vector< double > > **results**

4.3.1 Detailed Description

Creates an alphabet with every letter filled.

4.3.2 Member Function Documentation

4.3.2.1 createA()

```
vector< vector< double > > Filling::createA (
            double offsetY)
```

Creates a filled-in version of A.

Parameters

offsetY	the vertical offset for creating A

Returns

A vector with all the points needed to fill A

4.3.2.2 createB()

```
vector< vector< double > > Filling::createB (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of C.

Parameters

offsetX	the horizontal offset for creating B
offsetY	the vertical offset for creating B

Returns

A vector with all the points needed to fill B

4.3.2.3 createC()

```
vector< vector< double > > Filling::createC (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of C.

Parameters

offsetX	the horizontal offset for creating C
offsetY	the vertical offset for creating C

Returns

A vector with all the points needed to fill C

4.3.2.4 createD()

```
vector< vector< double > > Filling::createD (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of D.

Parameters

offsetX	the horizontal offset for creating D
offsetY	the vertical offset for creating D

Returns

A vector with all the points needed to fill D

4.3.2.5 createE()

```
vector< vector< double > > Filling::createE (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of E.

Parameters

offsetX	the horizontal offset for creating E
offsetY	the vertical offset for creating E

Returns

A vector with all the points needed to fill E

4.3.2.6 createF()

```
vector< vector< double > > Filling::createF (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of F.

Parameters

offsetX	the horizontal offset for creating F
offsetY	the vertical offset for creating F

Returns

A vector with all the points needed to fill F

4.3.2.7 createG()

```
vector< vector< double > > Filling::createG (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of G.

Parameters

offsetX	the horizontal offset for creating G
offsetY	the vertical offset for creating G

Returns

A vector with all the points needed to fill G

4.3.2.8 createH()

```
vector< vector< double > > Filling::createH (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of H.

Parameters

offsetX	the horizontal offset for creating H
offsetY	the vertical offset for creating H

Returns

A vector with all the points needed to fill H

4.3.2.9 createI()

```
vector< vector< double > > Filling::createI (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of I.

Parameters

offsetX	the horizontal offset for creating I
offsetY	the vertical offset for creating I

Returns

A vector with all the points needed to fill I

4.3.2.10 createJ()

```
vector< vector< double > > Filling::createJ (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of J.

Parameters

offsetX	the horizontal offset for creating J
offsetY	the vertical offset for creating J

Returns

A vector with all the points needed to fill J

4.3.2.11 createK()

```
vector< vector< double > > Filling::createK (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of K.

Parameters

offsetX	the horizontal offset for creating K
offsetY	the vertical offset for creating K

Returns

A vector with all the points needed to fill K

4.3.2.12 createL()

```
vector< vector< double > > Filling::createL (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of L.

Parameters

offsetX	the horizontal offset for creating L
offsetY	the vertical offset for creating L

Returns

A vector with all the points needed to fill L

4.3.2.13 createM()

```
vector< vector< double > > Filling::createM (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of M.

Parameters

offsetX	the horizontal offset for creating M
offsetY	the vertical offset for creating M

Returns

A vector with all the points needed to fill M

4.3.2.14 createN()

```
vector< vector< double > > Filling::createN (
            double offsetY)
```

Creates a filled-in version of N.

Parameters

offsetY	the vertical offset for creating N

Returns

A vector with all the points needed to fill N

4.3.2.15 createO()

```
vector< vector< double > > Filling::createO (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of O.

Parameters

offsetX	the horizontal offset for creating O
offsetY	the vertical offset for creating O

Returns

A vector with all the points needed to fill O

4.3.2.16 createP()

```
vector< vector< double > > Filling::createP (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of P.

Parameters

offsetX	the horizontal offset for creating P
offsetY	the vertical offset for creating P

Returns

A vector with all the points needed to fill P

4.3.2.17 createQ()

```
vector< vector< double > > Filling::createQ (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of Q.

Parameters

offsetX	the horizontal offset for creating Q
offsetY	the vertical offset for creating Q

Returns

A vector with all the points needed to fill Q

4.3.2.18 createR()

```
vector< vector< double > > Filling::createR (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of R.

Parameters

offsetX	the horizontal offset for creating R
offsetY	the vertical offset for creating R

Returns

A vector with all the points needed to fill R

4.3.2.19 createS()

```
vector< vector< double > > Filling::createS (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of S.

Parameters

offsetX	the horizontal offset for creating S
offsetY	the vertical offset for creating S

Returns

A vector with all the points needed to fill S

4.3.2.20 createT()

```
vector< vector< double > > Filling::createT (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of T.

Parameters

offsetX	the horizontal offset for creating T
offsetY	the vertical offset for creating T

Returns

A vector with all the points needed to fill T

4.3.2.21 createU()

```
vector< vector< double > > Filling::createU (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of U.

Parameters

offsetX	the horizontal offset for creating U
offsetY	the vertical offset for creating U

Returns

A vector with all the points needed to fill U

4.3.2.22 createV()

```
vector< vector< double > > Filling::createV (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of V.

Parameters

offsetX	the horizontal offset for creating V
offsetY	the vertical offset for creating V

Returns

A vector with all the points needed to fill V

4.3.2.23 createW()

```
vector< vector< double > > Filling::createW (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of W.

Parameters

offsetX	the horizontal offset for creating W
offsetY	the vertical offset for creating W

Returns

A vector with all the points needed to fill W

4.3.2.24 createX()

```
vector< vector< double > > Filling::createX (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of X.

Parameters

offsetX	the horizontal offset for creating X
offsetY	the vertical offset for creating X

Returns

A vector with all the points needed to fill X

4.3.2.25 createY()

```
vector< vector< double > > Filling::createY (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of Y.

Parameters

offsetX	the horizontal offset for creating Y
offsetY	the vertical offset for creating Y

Returns

A vector with all the points needed to fill Y

4.3.2.26 createZ()

```
vector< vector< double > > Filling::createZ (
            double offsetX,
            double offsetY)
```

Creates a filled-in version of Z.

Parameters

offsetX	the horizontal offset for creating Z
offsetY	the vertical offset for creating Z

Returns

A vector with all the points needed to fill Z

4.3.2.27 generateAlphabet()

```
void Filling::generateAlphabet (
            int x)  [override], [virtual]
```

Produces every point needed for a filled font.

Parameters

i	Distinguishes between the second and third alphabet

Implements Alphabet.

The documentation for this class was generated from the following files:

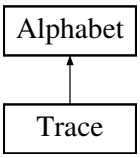
- Filling.h
- Filling.cpp

4.4 Trace Class Reference

Traces the outline of the letters of the alphabet.

`#include <Trace.h>`

Inheritance diagram for Trace:



Public Member Functions

- **Trace** ()=default

  Default Constructor for class Trace.
- void generateAlphabet (int x) override

  Produces all the points for the traced version of the alphabet.
- vector< vector< double > > createLetter (vector< vector< vector< double > > > letter, double offsetX, double offsetY)

  Creates a filled in version of the letter given.
- ~**Trace** ()=default

  Default Destructor for class Trace.

Public Member Functions inherited from Alphabet

- **Alphabet** ()=default

  Default Constructor for class Alphabet.
- virtual ~**Alphabet** ()

  Default Destructor for class Alphabet.

Additional Inherited Members

Public Attributes inherited from Alphabet

- vector< vector< vector< double > > > **letters**

Protected Attributes inherited from Alphabet

- vector< vector< double > > **results**

4.4.1 Detailed Description

Traces the outline of the letters of the alphabet.

4.4.2 Member Function Documentation

4.4.2.1 createLetter()

```
vector< vector< double > > Trace::createLetter (
            vector< vector< vector< double > > > letter,
            double offsetX,
            double offsetY)
```

Creates a filled in version of the letter given.

Parameters

letter	All the points for the DeCasteljau algorithm to create the lines of the letter
offsetX	The horizontal offset of the top left corner of the letter to be created
offsetY	The vertical offset of the top left corner of the letter to be created

Returns

all the points for all the lines of the letter

4.4.2.2 generateAlphabet()

```
void Trace::generateAlphabet (
            int x)  [override], [virtual]
```

Produces all the points for the traced version of the alphabet.

Parameters

x	Distinguishes between the first and third alphabet

Implements Alphabet.

The documentation for this class was generated from the following files:

- Trace.h
- Trace.cpp

Chapter 5

File Documentation

5.1 Alphabet.h

```
00001 #ifndef ALPHABET_H
00002 #define ALPHABET_H
00003
00004 #include <vector>
00005 #include "DeCasteljau.h"
00006 using namespace std;
00007
00012 struct Alphabet {
00013
00017     Alphabet() = default;
00018
00022     vector<vector<vector<double»> letters;
00027     virtual void generateAlphabet(int x) = 0;
00028
00032     virtual ~Alphabet() {};
00033
00034 protected:
00035     vector<vector<double» results;
00036 };
00037
00038 #endif // ALPHABET_H
```

5.2 DeCasteljau.h

```
00001 #ifndef DECASTELJAU_H
00002 #define DECASTELJAU_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <iostream>
00007 using namespace std;
00008
00013 class DeCasteljau {
00014     protected:
00022         static vector<double> interpolation(vector<double> p0, vector<double> p1, double t);
00023
00024     public:
00031         static vector<vector<double» linearBezier(vector<double> p0, vector <double> p1);
00032
00040         static vector<vector<double» quadraticBezier(vector <double> p0, vector <double> p1, vector
      <double> p2);
00041
00050         static vector<vector<double» cubicBezier(vector <double> p0, vector <double> p1, vector
      <double> p2, vector <double> p3);
00051 };
00052
00053 #endif //DECASTELJAU_H
```

5.3 Filling.h

```
00001 #ifndef FILLING_H
00002 #define FILLING_H
00003
00004 #include "Alphabet.h"
00005
00010 class Filling : public Alphabet {
00011
00012     vector<vector<double» coordinates = {{190.0, 270.0}, {370.0, 450.0}};
00013
00014 public:
00019     void generateAlphabet(int x) override;
00020
00026     vector<vector<double» createA(double offsetY);
00027
00034     vector<vector<double» createB(double offsetX, double offsetY);
00035
00042     vector<vector<double» createC(double offsetX, double offsetY);
00043
00050     vector<vector<double» createD(double offsetX, double offsetY);
00051
00058     vector<vector<double» createE(double offsetX, double offsetY);
00059
00066     vector<vector<double» createF(double offsetX, double offsetY);
00067
00074     vector<vector<double» createG(double offsetX, double offsetY);
00075
00082     vector<vector<double» createH(double offsetX, double offsetY);
00083
00090     vector<vector<double» createI(double offsetX, double offsetY);
00091
00098     vector<vector<double» createJ(double offsetX, double offsetY);
00099
00106     vector<vector<double» createK(double offsetX, double offsetY);
00107
00114     vector<vector<double» createL(double offsetX, double offsetY);
00115
00122     vector<vector<double» createM(double offsetX, double offsetY);
00123
00129     vector<vector<double» createN(double offsetY);
00130
00137     vector<vector<double» createO(double offsetX, double offsetY);
00138
00145     vector<vector<double» createP(double offsetX, double offsetY);
00146
00153     vector<vector<double» createQ(double offsetX, double offsetY);
00154
00161     vector<vector<double» createR(double offsetX, double offsetY);
00162
00169     vector<vector<double» createS(double offsetX, double offsetY);
00170
00177     vector<vector<double» createT(double offsetX, double offsetY);
00178
00185     vector<vector<double» createU(double offsetX, double offsetY);
00186
00193     vector<vector<double» createV(double offsetX, double offsetY);
00194
00201     vector<vector<double» createW(double offsetX, double offsetY);
00202
00209     vector<vector<double» createX(double offsetX, double offsetY);
00210
00217     vector<vector<double» createY(double offsetX, double offsetY);
00218
00225     vector<vector<double» createZ(double offsetX, double offsetY);
00226
00230     ~Filling() override = default;
00231 };
00232
00233 #endif //FILLING_H
```

5.4 Sdl.h File Reference

Header file containing functions for SDL initialization, rendering, and cleanup.

```
#include <SDL2/SDL.h>
#include <iostream>
#include <vector>
#include "DeCasteljau.h"
```

Functions

- bool init (SDL_Window **window, SDL_Renderer **renderer)
  Initializes SDL, creates a window, and a renderer.
- void firstAlphabet (SDL_Renderer **renderer, vector< vector< vector< double > > > letters)
  Draws the first alphabet using the provided letters.
- void draw (SDL_Renderer **renderer, vector< vector< vector< double > > > letters)
  Draws the provided letters to the screen.
- void drawSeparator (SDL_Renderer **renderer, vector< double > start, vector< double > finish)
  Draws a separator line between two points using linear Bézier interpolation.
- void wait (SDL_Renderer **renderer)
  Waits for a quit event to close the window and renderer.
- void close (SDL_Window **window, SDL_Renderer **renderer)
  Cleans up SDL resources and closes the window and renderer.

5.4.1 Detailed Description

Header file containing functions for SDL initialization, rendering, and cleanup.

5.4.2 Function Documentation

5.4.2.1 close()

```
void close (
            SDL_Window ** window,
            SDL_Renderer ** renderer)
```

Cleans up SDL resources and closes the window and renderer.

Parameters

window	A pointer to the SDL_Window to be destroyed.
renderer	A pointer to the SDL_Renderer to be destroyed.

5.4.2.2 draw()

```
void draw (
            SDL_Renderer ** renderer,
            vector< vector< vector< double > > > letters)
```

Draws the provided letters to the screen.

Parameters

renderer	A pointer to the SDL_Renderer used to draw the points.
letters	A 3D vector containing the points for all letters.

5.4.2.3 drawSeparator()

```
void drawSeparator (
            SDL_Renderer ** renderer,
            vector< double > start,
            vector< double > finish)
```

Draws a separator line between two points using linear Bézier interpolation.

Parameters

renderer	A pointer to the SDL_Renderer used to draw the line.
start	A vector containing the x and y coordinates of the starting point.
finish	A vector containing the x and y coordinates of the finishing point.

5.4.2.4 firstAlphabet()

```
void firstAlphabet (
            SDL_Renderer ** renderer,
            vector< vector< vector< double > > > letters)
```

Draws the first alphabet using the provided letters.

Parameters

renderer	A pointer to the SDL_Renderer used to draw the points.
letters	A 3D vector containing the points for all letters.

5.4.2.5 init()

```
bool init (
            SDL_Window ** window,
            SDL_Renderer ** renderer)
```

Initializes SDL, creates a window, and a renderer.

Parameters

window	A pointer to the SDL_Window object that will be created.
renderer	A pointer to the SDL_Renderer object that will be created.

Returns

True if initialization succeeds, false otherwise.

5.4.2.6 wait()

```
void wait (
            SDL_Renderer ** renderer)
```

Waits for a quit event to close the window and renderer.

Parameters

renderer	A pointer to the SDL_Renderer used to present the content.

5.5 Sdl.h

Go to the documentation of this file.

```
00001
00005 #ifndef SDL_H
00006 #define SDL_H
00007
00008 #include <SDL2/SDL.h>
00009 #include <iostream>
00010 #include <vector>
00011 #include "DeCasteljau.h"
00012 using namespace std;
00013
00022 bool init(SDL_Window** window, SDL_Renderer** renderer);
00023
00030 void firstAlphabet(SDL_Renderer** renderer, vector<vector<vector<double>> letters);
00031
00038 void draw(SDL_Renderer** renderer, vector<vector<vector<double>> letters);
00039
00047 void drawSeparator(SDL_Renderer** renderer, vector<double> start, vector<double> finish);
00048
00054 void wait(SDL_Renderer** renderer);
00055
00062 void close(SDL_Window** window, SDL_Renderer** renderer);
00063
00064 #endif // SDL_H
```

5.6 Trace.h

```
00001 #ifndef TRACE_H
00002 #define TRACE_H
00003
00004 #include "Alphabet.h"
00005
00010 class Trace : public Alphabet{
00014     vector<vector<vector<double>> A1 = { {{15,75}, {40,5}}, {{55,5}, {80,75}}, {{35,55}, {60,55}},
      {{40,5}, {55,5}}, {{15,75}, {30,75}}, {{65,75}, {80,75}}, {{30,75}, {35,55}}, {{60,55}, {65,75}},
      {{37,45}, {58,45}}, {{37,45}, {48,20}}, {{48,20}, {58, 45}} };
00015     vector<vector<vector<double>> B1= { {{15,5}, {15,75}}, {{30,17}, {30,29}}, {{30,52}, {30,64}},
      {{15,5}, {80,0}, {80,40}, {30,40}}, {{30,40}, {80,35}, {80,80}, {15,75}}, {{30,17}, {55,15}, {55,30},
      {30,29}}, {{30,52}, {55,52}, {55,67}, {30,64}} };
00016     vector<vector<vector<double>> C1 = { {{50,5},{50,20}}, {{50,60},{50,75}},{{50,5},
      {-15,0},{-15,80},{50,75}}, {{50,20},{0,15},{0,65},{50,60}} };
00017     vector<vector<vector<double>> D1 = { {{20,5},{20,75}}, {{20,5},{80,-5},{80,85},{20,75}},
      {{35,20},{35,60}}, {{35,20},{65,10},{65,70},{35,60}} };
00018     vector<vector<vector<double>> E1 = { {{15,5},{15,75}}, {{15,5},{60,5}}, {{60,5},{60,20}},
      {{60,20},{30,20}}, {{30,20},{30,32}}, {{30,32},{55,32}}, {{55,32},{55,47}}, {{55,47},{30,47}},
      {{30,47}, {30,60}}, {{30,60},{60,60}}, {{60,60}, {60,75}}, {{60,75}, {15,75}} };
00019     vector<vector<vector<double>> F1 = { {{15,5},{15,75}}, {{15,5},{60,5}}, {{60,5},{60,20}},
      {{60,20},{30,20}}, {{30,20},{30,32}}, {{30,32},{45,32}}, {{45,32},{45,47}}, {{45,47},{30,47}},
      {{30,47},{30,75}}, {{30,75},{15,75}} };
00020     vector<vector<vector<double>> G1 = { {{60,5},{60,20}}, {{60,5},{-20,-15},{-15,95},{60,70}},
      {{60,20},{-5,0},{0,70},{45,55}}, {{60,70},{60,35}}, {{45,55},{45,50}}, {{45,50},{30,50}},
      {{60,35},{30,35}}, {{30,35},{30,50}} };
00021     vector<vector<vector<double>> H1 = { {{15,5},{30,5}}, {{30,5},{30,32}}, {{30,32},{45,32}},
      {{45,32},{45,5}}, {{45,5},{60,5}}, {{60,5},{60,75}}, {{60,75},{45,75}}, {{45,75},{45,47}},
      {{45,47},{30,47}}, {{30,47},{30,75}}, {{30,75},{15,75}}, {{15,75},{15,5}} };
00022     vector<vector<vector<double>> I1 = { {{15,5},{50,5}}, {{50,5},{50,20}}, {{50,20},{40,20}},
      {{40,20},{40,60}}, {{40,60},{50,60}}, {{50,60},{50,75}}, {{50,75},{15,75}}, {{15,75},{15,60}},
      {{15,60},{26,60}}, {{27,60},{27,20}}, {{26,20},{15,20}}, {{15,20},{15,5}} };
00023     vector<vector<vector<double>> J1 = { {{25,5},{70,5}}, {{25,5},{25,20}}, {{25,20},{40,20}},
      {{54,20},{70,20}}, {{70,20},{70,5}}, {{40,20},{40,57}}, {{54,20},{54,57}},
      {{54,57},{54,80},{15,80},{15,57}}, {{40,57},{40,66},{29,66},{29,57}}, {{15,57},{29,57}} };
00024     vector<vector<vector<double>> K1 = { {{15,5},{30,5}}, {{30,5},{30,32}}, {{30,32},{55,5}},
      {{55,5},{70,5}}, {{70,5},{40,39}}, {{40,39},{70,75}}, {{55,75},{70,75}}, {{ 55,75},{30,47}},
      {{30,47},{30,75}}, {{15,75},{30,75}}, {{15,75},{15,5}} };
00025     vector<vector<vector<double>> L1 = { {{15,5},{15,75}}, {{15,5},{30,5}}, {{30,5},{30,60}},
      {{30,60},{60,60}}, {{60,60},{60,75}}, {{60,75},{15,75}} };
00026     vector<vector<vector<double>> M1 = { {{15,5},{15,75}}, {{30,5},{15,5}}, {{30,5},{40,25}},
      {{40,25},{50,5}}, {{50,5},{65,5}}, {{65,5},{65,75}}, {{50,75},{65,75}}, {{50,75},{50, 30}},
      {{50,30},{40,45}}, {{40,45},{30,30}}, {{30,30},{30,75}}, {{30,75},{15,75}} };
00027     vector<vector<vector<double>> N1 = { {{15,15},{15,85}}, {{15,15},{30,15}}, {{30,15},{65,65}},
      {{65,65},{65,15}}, {{65,15},{80,15}}, {{80,15},{80,85}}, {{65,85},{80,85}}, {{65,85},{30,35}},
      {{30,35},{30,85}}, {{15,85},{30,85}} };
00028     vector<vector<vector<double>> O1 = { {{50,15},{0,10},{0,90},{50,85}},
      {{50,15},{100,10},{100,90},{50,85}}, {{50,30},{20,30},{20,75},{50,70}},
      {{50,30},{80,30},{80,75},{50,70}} };
00029     vector<vector<vector<double>> P1 = { {{30,15},{30,85}}, {{45,50},{45,85}}, {{45,85},{30,85}},
      {{45,25},{45,40}}, {{30,15},{80,5},{80,55},{45,50}}, {{45,25},{60,20},{60,45},{45,40}} };
00030     vector<vector<vector<double>> Q1 = { {{40,15},{5,10},{0,85},{40,85}},
      {{40,30},{20,25},{15,70},{40,70}}, {{55,75},{65,85}}, {{60,67},{70,75}}, {{70,75},{65,85}},
      {{40,15},{70,10},{70,85},{40,85}}, {{40,30},{55,25},{55,70},{40,70}}, {}, {{56,75},{59,70}} };
```

```
00031     vector<vector<vector<double>> R1 = { {{15,15},{15,85}}, {{30,50},{60,85}}, {{60,85},{45,85}},
      {{45,85},{30,65}}, {{30,65},{30,85}}, {{30,85},{15,85}}, {{15,85},{15,15}}, {{30,25},{30,40}},
      {{15,15},{65,5},{65,55},{30,50}}, {{30,25},{45,20},{45,45},{30,40}} };
00032     vector<vector<vector<double>> S1 = { {{50,30},{55,20},{15,25},{35,45}},
      {{50,15},{15,5},{10,57},{35,52}}, {{50,30},{50,15}}, {{20,70},{20,85}},
      {{35,45},{60,45},{60,90},{20,85}}, {{35,52},{45,60},{45,80},{20,70}} };
00033     vector<vector<vector<double>> T1 = { {{5,15},{70,15}}, {{5,15},{5,30}}, {{5,30},{30,30}},
      {{44,30},{70,30}}, {{70,30},{70,15}}, {{30,30},{30,85}}, {{44,30},{44,85}}, {{44,85},{30,85}} };
00034     vector<vector<vector<double>> U1 = { {{15,15},{30,15}}, {{15,15},{15,60}}, {{30,15},{30,65}},
      {{60,60},{60,15}}, {{60,15},{75,15}}, {{75,15},{75,60}}, {{30,60},{40,78},{50,78},{60,60}},
      {{15,60},{25,92},{65,92},{75,60}} };
00035     vector<vector<vector<double>> V1 = { {{15,15},{30,15}}, {{60,15},{75,15}}, {{30,15},{45,70}},
      {{45,70},{60,15}}, {{15,15},{38,85}}, {{38,85},{52,85}}, {{52,85},{75,15}} };
00036     vector<vector<vector<double>> W1 = { {{15,15},{30,15}}, {{60,15},{75,15}}, {{30,15},{33,65}},
      {{33,65},{40,50}}, {{40,50},{50,50}}, {{50,50},{57,65}}, {{57,65},{60,15}}, {{75,15},{65,85}},
      {{65,85},{50,85}}, {{50,85},{45,65}}, {{45,65},{40,85}}, {{40,85},{25,85}}, {{25,85},{15,15}} };
00037     vector<vector<vector<double>> X1 = { {{15,15},{30,15}}, {{60,15},{75,15}}, {{15,85},{30,85}},
      {{60,85},{75,85}}, {{30,15},{45,40}}, {{45,40},{60,15}}, {{75,15},{55,50}}, {{55,50},{75,85}},
      {{60,85},{45,60}}, {{45,60},{30,85}}, {{15,85},{35,50}}, {{35,50},{15,15}} };
00038     vector<vector<vector<double>> Y1 = { {{15,15},{30,15}}, {{60,15},{75,15}}, {{30,15},{45,40}},
      {{45,40},{60,15}}, {{75,15},{52,53}}, {{15,15},{38,53}}, {{38,53},{38,85}}, {{52,53},{52,85}},
      {{38,85},{52,85}} };
00039     vector<vector<vector<double>> Z1 = { {{15,15},{80,15}}, {{15,30},{65,30}}, {{15,85},{80,85}},
      {{30,70},{80,70}}, {{15,85},{15,70}}, {{80,15},{80,30}}, {{80,30},{30,70}}, {{65,30},{15,70}},
      {{15,15},{15,30}}, {{80,85},{80,70}} };
00043     vector<vector<vector<double>> A3 = {
      {{12,77},{38,3}},{{56,3},{82,77}},{{37,61},{58,61}},{{38,3},{57,3}},{{13,77},{30,77}},{{63,77},{83,77}},{{30,77},{37,61
      };
00044     vector<vector<vector<double>> B3 = { {{13, 3}, {13, 77}}, {{32, 19}, {32, 27}}, {{32, 54}, {31,
      62}}, {{12, 3}, {82, -4}, {82, 40}, {37, 42}}, {{38, 41}, {82, 32}, {82, 83}, {13, 77}}, {{32, 18},
      {51, 16}, {51, 27}, {31, 27}}, {{32, 54}, {50, 51}, {50, 67}, {31, 62}} };
00045     vector<vector<vector<double>> C3 = { {{52, 3}, {52, 22}}, {{52, 58}, {52, 77}}, {{52, 3}, {-18,
      -4}, {-18, 83}, {52, 77}}, {{52, 22}, {2, 12}, {2, 67}, {52, 58}} };
00046     vector<vector<vector<double>> D3 = { {{18, 3}, {18, 77}}, {{18, 3}, {83, -9}, {83, 87}, {18, 77}},
      {{36, 21}, {36, 59}}, {{36, 21}, {62, 8}, {62, 71}, {36, 59}} };
00047     vector<vector<vector<double>> E3 = { {{13, 3}, {13, 77}}, {{13, 3}, {62, 3}}, {{62, 3}, {62, 22}},
      {{62, 21}, {31, 21}}, {{31, 21}, {31, 30}}, {{32, 30}, {57, 30}}, {{57, 30}, {57, 49}}, {{57, 48},
      {31, 48}}, {{31, 48}, {31, 58}}, {{32, 58}, {62, 58}}, {{62, 58}, {62, 77}}, {{62, 76}, {13, 76}} };
00048     vector<vector<vector<double>> F3 = { {{13, 3}, {13, 77}}, {{13, 3}, {62, 3}}, {{62, 3}, {62, 22}},
      {{62, 21}, {32, 21}}, {{31, 22}, {31, 32}}, {{32, 31}, {47, 31}}, {{47, 34}, {47, 49}}, {{47, 48},
      {32, 48}}, {{31, 48}, {31, 77}}, {{32, 77}, {13, 77}} };
00049     vector<vector<vector<double>> G3 = { {{62, 3}, {62, 22}}, {{62, 3}, {-24, -18}, {-19, 98}, {62,
      72}}, {{62, 23}, {-3, 2}, {2, 66}, {46, 53}}, {{62, 72}, {62, 33}}, {{46, 52}, {28, 52}}, {{63, 33},
      {28, 33}}, {{28, 33}, {28, 52}} };
00050     vector<vector<vector<double>> H3 = { {{13, 3}, {31, 3}}, {{31, 3}, {31, 30}}, {{32, 30}, {43,
      30}}, {{43, 30}, {43, 3}}, {{43, 3}, {62, 3}}, {{61, 3}, {61, 77}}, {{62, 77}, {43, 77}}, {{43, 77},
      {43, 49}}, {{43, 49}, {32, 49}}, {{31, 49}, {31, 77}}, {{31, 77}, {13, 77}}, {{13, 77}, {13, 3}} };
00051     vector<vector<vector<double>> I3 = { {{13, 3}, {52, 3}}, {{52, 3}, {52, 22}}, {{52, 21}, {41,
      21}}, {{41, 22}, {41, 58}}, {{41, 58}, {52, 58}}, {{52, 58}, {52, 76}}, {{52, 76}, {13, 76}}, {{13,
      76}, {13, 58}}, {{13, 58}, {23, 58}}, {{23, 58}, {23, 22}}, {{23, 21}, {13, 21}}, {{13, 22}, {13, 3}}
      };
00052     vector<vector<vector<double>> J3 = { {{23, 3}, {72, 3}}, {{23, 3}, {23, 21}}, {{23, 20}, {38,
      20}}, {{56, 20}, {72, 20}}, {{72, 21}, {72, 3}}, {{38, 22}, {38, 57}}, {{55, 22}, {55, 57}}, {{56,
      57}, {55, 83}, {15, 83}, {13, 55}}, {{38, 57}, {40, 64}, {29, 64}, {31, 55}}, {{13, 55}, {31, 55}} };
00053     vector<vector<vector<double>> K3 = { {{13, 3}, {31, 3}}, {{31, 3}, {31, 30}}, {{32, 30}, {53, 3}},
      {{53, 3}, {73, 3}}, {{73, 3}, {42, 39}}, {{42, 39}, {73, 77}}, {{73, 77}, {53, 77}}, {{53, 77}, {31,
      49}}, {{31, 49}, {31, 77}}, {{31, 77}, {13, 77}}, {{13, 77}, {13, 3}} };
00054     vector<vector<vector<double>> L3 = { {{13, 3}, {13, 77}}, {{13, 3}, {32, 3}}, {{31, 3}, {31, 58}},
      {{32, 58}, {62, 58}}, {{62, 58}, {62, 77}}, {{62, 76}, {13, 76}} };
00055     vector<vector<vector<double>> M3 = { {{14, 3}, {14, 77}}, {{13, 3}, {32, 3}}, {{32, 3}, {40, 21}},
      {{40, 21}, {48, 3}}, {{48, 3}, {66, 3}}, {{66, 3}, {66, 77}}, {{66, 77}, {48, 77}}, {{48, 77}, {48,
      29}}, {{48, 29}, {40, 44}}, {{40, 44}, {32, 30}}, {{32, 32}, {32, 77}}, {{32, 77}, {13, 77}} };
00056     vector<vector<vector<double>> N3 = { {{13, 13}, {13, 87}}, {{13, 13}, {31, 13}}, {{31, 13}, {64,
      62}}, {{63, 63}, {63, 13}}, {{63, 13}, {81, 13}}, {{81, 13}, {81, 87}}, {{81, 87}, {63, 87}}, {{64,
      87}, {31, 39}}, {{31, 37}, {31, 87}}, {{31, 87}, {13, 87}} };
00057     vector<vector<vector<double>> O3 = { {{50, 13}, {-3, 9}, {-3, 90}, {48, 88}}, {{50, 13}, {103, 9},
      {103, 91}, {48, 88}}, {{50, 31}, {22, 33}, {22, 72}, {50, 68}}, {{50, 31}, {77, 33}, {77, 72}, {50,
      68}} };
00058     vector<vector<vector<double>> P3 = { {{29, 13}, {29, 87}}, {{29, 13}, {82, 2}, {82, 58}, {47,
      56}}, {{47, 52}, {47, 87}}, {{47, 87}, {28, 87}}, {{47, 27}, {47, 38}}, {{47, 27}, {62, 25}, {61, 41},
      {47, 37}} };
00059     vector<vector<vector<double>> Q3 = { {{40, 14}, {3, 8}, {-2, 87}, {40, 86}}, {{40, 31}, {24, 27},
      {17, 68}, {40, 68}}, {{55, 78}, {65, 88}}, {{60, 63}, {72, 73}}, {{73, 73}, {67, 87}}, {{40, 14}, {72,
      8}, {72, 87}, {40, 86}}, {{40, 31}, {52, 27}, {52, 68}, {40, 68}} };
00060     vector<vector<vector<double>> R3 = { {{13, 13}, {13, 87}}, {{13, 13}, {68, -1}, {67, 57}, {35,
      53}}, {{36, 54}, {65, 87}}, {{64, 87}, {44, 87}}, {{44, 87}, {30, 67}}, {{31, 67}, {31, 87}}, {{31,
      87}, {13, 87}}, {{13, 87}, {13, 13}}, {{31, 27}, {31, 38}}, {{29, 30}, {48, 20}, {48, 39}, {29, 36}}
      };
00061     vector<vector<vector<double>> S3 = { {{52, 33}, {43, 21}, {25, 30}, {35, 42}}, {{52, 13}, {12, 2},
      {7, 58}, {35, 55}}, {{52, 33}, {52, 13}}, {{18, 67}, {18, 87}}, {{35, 43}, {62, 43}, {63, 92}, {18,
      87}}, {{35, 55}, {43, 62}, {43, 78}, {18, 67}} };
00062     vector<vector<vector<double>> T3 = { {{3, 13}, {72, 13}}, {{3, 13}, {3, 31}}, {{3, 31}, {28, 31}},
      {{46, 31}, {72, 31}}, {{72, 31}, {72, 13}}, {{28, 31}, {28, 87}}, {{46, 31}, {46, 87}}, {{46, 87},
      {28, 87}} };
00063     vector<vector<vector<double>> U3 = { {{13, 13}, {32, 13}}, {{13, 13}, {13, 62}}, {{31, 13}, {31,
      62}}, {{58, 62}, {58, 13}}, {{58, 13}, {76, 13}}, {{76, 13}, {76, 62}}, {{32, 62}, {39, 74}, {51, 74},
      {58, 62}}, {{13, 62}, {23, 94}, {67, 94}, {77, 62}} };
```

```
00064     vector<vector<vector<double>> V3 = { {{12, 13}, {31, 13}}, {{61, 13}, {77, 13}}, {{31, 13}, {44,
      57}}, {{44, 57}, {58, 13}}, {{12, 13}, {36, 87}}, {{36, 87}, {54, 87}}, {{54, 87}, {77, 13}} };
00065     vector<vector<vector<double>> W3 = { {{13, 13}, {31, 13}}, {{58, 13}, {77, 13}}, {{31, 13}, {38,
      63}}, {{38, 63}, {39, 48}}, {{39, 48}, {51, 48}}, {{51, 48}, {56, 63}}, {{52, 63}, {58, 13}}, {{76,
      13}, {66, 87}}, {{67, 87}, {48, 87}}, {{48, 87}, {45, 70}}, {{45, 70}, {42, 87}}, {{40, 87}, {22,
      87}}, {{23, 87}, {13, 13}} };
00066     vector<vector<vector<double>> X3 = { {{13, 13}, {31, 13}}, {{58, 13}, {77, 13}}, {{13, 87}, {31,
      87}}, {{58, 87}, {77, 87}}, {{31, 13}, {44, 35}}, {{44, 35}, {58, 13}}, {{77, 13}, {55, 50}}, {{55,
      50}, {77, 87}}, {{58, 87}, {44, 65}}, {{44, 65}, {31, 87}}, {{13, 87}, {34, 50}}, {{34, 50}, {13, 13}}
      };
00067     vector<vector<vector<double>> Y3 = { {{13, 13}, {31, 13}}, {{58, 13}, {77, 13}}, {{31, 13}, {44,
      38}}, {{44, 38}, {58, 13}}, {{78, 13}, {55, 53}}, {{12, 13}, {35, 53}}, {{36, 53}, {36, 87}}, {{54,
      53}, {54, 87}}, {{36, 87}, {54, 87}} };
00068     vector<vector<vector<double>> Z3 = { {{13, 13}, {82, 13}}, {{13, 31}, {58, 31}}, {{13, 86}, {82,
      86}}, {{36, 68}, {82, 68}}, {{13, 87}, {13, 68}}, {{82, 13}, {82, 31}}, {{82, 31}, {37, 68}}, {{58,
      32}, {13, 68}}, {{13, 13}, {13, 32}}, {{82, 87}, {82, 68}} };
00069
00070     vector<vector<vector<vector<vector<double>>>*>> ab = { {&A1, &B1, &C1, &D1, &E1, &F1, &G1, &H1, &I1,
      &J1, &K1, &L1, &M1, &N1, &O1, &P1, &Q1, &R1, &S1, &T1, &U1, &V1, &W1, &X1, &Y1, &Z1},
00071                                                             {&A3, &B3, &C3, &D3, &E3, &F3, &G3, &H3,
      &I3, &J3, &K3, &L3, &M3, &N3, &O3, &P3, &Q3, &R3, &S3, &T3, &U3, &V3, &W3, &X3, &Y3, &Z3} };
00075     vector<vector<double> coordinates = { {20.0, 90.0}, {370.0, 450.0} };
00076
00077     vector<vector<double> line;
00078
00079 public:
00080
00084     Trace() = default;
00085
00086
00091     void generateAlphabet(int x) override;
00092
00100     vector<vector<double> createLetter(vector<vector<vector<double>> letter, double offsetX, double
      offsetY);
00101
00105     ~Trace() = default;
00106 };
00107
00108 #endif //TRACE_H
```

Index