Aztec Code generator

Description

The library is intended for creation "Aztec Code" barcodes. It supports C and Delphi interfaces and is available under the "Modified BSD License" (the text is in the license.txt file).

Types

ag_matrix

The structure represents a matrix of points describes a barcode (0 is a white point, 1 is a black point).

```
typedef struct ag_matrix_tag
{
	unsigned char *data;
	size_t width;
	size_t height;
}
ag_matrix;
```

Members:

Data	A pointer to the array of points. "Row-major" order is used, rows aren't aligned
Width	The number of columns
Height	The number of rows

ag_settings

The structure contains settings for generation of a matrix of points.

```
typedef struct ag_settings_tag
{
	unsigned long mask;
	unsigned char symbol_type;
	unsigned char symbol_format;
	unsigned char redundancy_for_error_correction;
}
ag_settings;
```

Members:

mask	A set of flags that specify which members of this structure contain valid data. This parameter can be one or more of the `AG_SF_...` values
symbol_type	The symbol's type. This parameter must be one of the `AG_..._SYMBOL` values (by default is `AG_NORMAL_SYMBOL`)
symbol_format	The symbol's format. This parameter must be one of the `AG_..._FORMAT` values (by default is `AG_ANYONE_FORMAT`)
redundancy_for_error_correction	The symbol's part (%) is allocated for error correction words. This parameter must be a value from 0 to 99 (by default is 23)

Functions

ag_generate

The function generates a matrix of points.

```
AG_API(int) ag_generate(ag_matrix **matrix, const void *data,
      size_t data_size, const ag_settings *settings);
```

Parameters:

Matrix	A pointer to a pointer to the generated matrix of points
Data	A pointer to a buffer containing the data to be encoded
data_size	The number of bytes to be encoded
Settings	A pointer to the structure with settings for generation. If the pointer is NULL, default settings are used

Return value:

See Error codes.

ag_release_matrix

The function deallocates a memory allocated for a matrix of points.

```
AG_API(int) ag_release_matrix(ag_matrix *matrix);
```

Parameters:

Matrix	A pointer to the released matrix of points

Return value:

See Error codes.

Error codes

Name	Code	Description
AG_SUCCESS	0	The operation completed successfully
AG_INVALID_PARAMETER	1	The input parameter is invalid
AG_OUT_OF_MEMORY	2	Not enough storage is available to complete this operation
AG_UNHANDLED_EXCEPTION	3	An unknown error occurred
AG_NO_INPUT_DATA	4	The function called without input data
AG_TOO_MUCH_INPUT_DATA	5	The function called with too much input data

Usage example

Source code:

```
#include <cstddef>
#include <cstdio>
#include "aztecgen.h"


int main()
{
	enum {success, error};

	const char data[] = "Hello world :)";
	enum {symbol_format = AG_FULL_FORMAT};

	// Barcode creation
	ag_settings settings;
	settings.mask = AG_SF_SYMBOL_FORMAT;
	settings.symbol_format = symbol_format;

	ag_matrix *barcode;
	const int gen_result = ag_generate(
		&barcode, data, sizeof(data) - 1, &settings);
	if (gen_result != AG_SUCCESS)
	{
		printf("Can't create a barcode (error code: %i)\n",
			gen_result);
		return error;
	}

	// Barcode printout
	printf("Data:\n");
	printf("\t%s\n", data);

	printf("Barcode:\n");
	for (size_t y = 0; y < barcode->height; ++y)
	{
		printf("\t");
		for (size_t x = 0; x < barcode->width; ++x)
		{
			const unsigned char t =
				barcode->data[y * barcode->width + x];
			printf("%c", (t == 0) ? ' ' : '#');
		}
		printf("\n");
	}
	printf("\n");

	// Barcode destruction
	ag_release_matrix(barcode);

	return success;
}
```

Output:

```
Data:
        Hello world :)
Barcode:
        # # ##      # # ### #
          ### #  ####### #### #
         # # ## ###   ## # # #
        ### # ###### ##   ### #
          #      ##    ##   ###
        #  # ############## #
         ### #            ##  #
        ## # # ######### #### #
        # #### #       # #   #
        #  # # # ##### # #
           # # # #   # # # #
         # # # # # # # # # # #
        #### # # #   # # #    #
           ### # ##### # ####
           # # #       # #
         # ### ######### ####
        # ####           #   ##
            ################# #
        ### ##    #       #   #
         ### ### #### # ##  #
        # # ##  ##   #   ## # #
        #### #   # ###########
        #### #  # #   # # ### #
```