
SGG Drawing Utility Functions

Επιμέλεια: Α.Δρακόπουλος

Πίνακας περιεχομένων

Βιβλιοθήκη διδιάστατων σχημάτων.....	3
Geometry_Shapes_H.....	3
Πρόγραμμα χρήσης της Geometry_Shapes.h.....	7
Επεξήγηση.....	8
Πρόγραμμα χρήσης των Geometry_Shapes.h , brush_utils.h.....	9
Πρόγραμμα.....	9
Επεξήγηση.....	10
Βιβλιοθήκη τρισδιάστατων σχημάτων.....	11
Βιβλιοθήκη three_d_shapes.h.....	11
Επεξήγηση.....	14
Πρόγραμμα χρήσης της "three_d_shapes.h".....	15
Επεξήγηση Προγράμματος.....	16
Παρατηρήσεις.....	16
Πρόγραμμα χρήσης της "three_d_shapes.h", "brush_utils.h".....	17
Επεξηγήσεις.....	18

Βιβλιοθήκη διδιάστατων σχημάτων

Geometry_Shapes_H

Ακολουθεί η υλοποίηση της βιβλιοθήκης διδιάστατων γεωμετρικών σχημάτων. Το αρχείο `geometry_shapes.h` περιέχει τις συναρτήσεις σχεδίασης με πλήρη documentation και σχόλια.

```
#pragma once
#ifndef GEOMETRY_SHAPES_H
#define GEOMETRY_SHAPES_H

#include "sgg/graphics.h"
#include <cmath>
#define M_PI 3.14159265358979323846
/**
 * Το namespace GeometryShapes περιέχει συναρτήσεις για τη σχεδίαση
 * βασικών διδιάστατων γεωμετρικών σχημάτων.
 */
namespace GeometryShapes {

    /**
     * Σχεδιάζει ένα τετράγωνο.
     * @param x Η x συντεταγμένη του κέντρου.
     * @param y Η y συντεταγμένη του κέντρου.
     * @param side Το μήκος της πλευράς.
     * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
     */
    void drawSquare(float x, float y, float side, const graphics::Brush& brush) {
        graphics::drawRect(x, y, side, side, brush);
    }

    /**
     * Σχεδιάζει ένα παραλληλόγραμμο.
     * @param x Η x συντεταγμένη του κέντρου.
     * @param y Η y συντεταγμένη του κέντρου.
     * @param width Το πλάτος του παραλληλογράμμου.
     * @param height Το ύψος του παραλληλογράμμου.
     * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
     */
    void drawRectangle(float x, float y, float width, float height, const
graphics::Brush& brush) {
        graphics::drawRect(x, y, width, height, brush);
    }

    /**
     * Σχεδιάζει ένα ρόμβο.
     * @param x Η x συντεταγμένη του κέντρου.
     * @param y Η y συντεταγμένη του κέντρου.
     * @param diagonal1 Η πρώτη διαγώνιος του ρόμβου.
     * @param diagonal2 Η δεύτερη διαγώνιος του ρόμβου.
     * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
     */
    void drawRhombus(float x, float y, float diagonal1, float diagonal2, const
graphics::Brush& brush) {
        float halfD1 = diagonal1 / 2;
        float halfD2 = diagonal2 / 2;

        graphics::drawLine(x, y - halfD1, x + halfD2, y, brush); // επάνω δεξιά
        graphics::drawLine(x + halfD2, y, x, y + halfD1, brush); // κάτω δεξιά
        graphics::drawLine(x, y + halfD1, x - halfD2, y, brush); // κάτω αριστερά
        graphics::drawLine(x - halfD2, y, x, y - halfD1, brush); // επάνω αριστερά
    }

}
```

```

}

/**
 * Σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο.
 * @param x H x συντεταγμένη του κέντρου.
 * @param y H y συντεταγμένη του κέντρου.
 * @param width Το πλάτος του ορθογώνιου παραλληλογράμμου.
 * @param height Το ύψος του ορθογώνιου παραλληλογράμμου.
 * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
 */
void drawParallelogram(float x, float y, float width, float height, const
graphics::Brush& brush) {
    float offsetX = width / 4;

    graphics::drawLine(x - offsetX, y - height / 2, x + width - offsetX, y -
height / 2, brush); // πάνω γραμμή
    graphics::drawLine(x + width - offsetX, y - height / 2, x + offsetX, y +
height / 2, brush); // δεξιά γραμμή
    graphics::drawLine(x + offsetX, y + height / 2, x - width + offsetX, y +
height / 2, brush); // κάτω γραμμή
    graphics::drawLine(x - width + offsetX, y + height / 2, x - offsetX, y -
height / 2, brush); // αριστερή γραμμή
}

/**
 * Σχεδιάζει ένα τραπέζιο.
 * @param x H x συντεταγμένη του κέντρου.
 * @param y H y συντεταγμένη του κέντρου.
 * @param topBase Το μήκος της πάνω βάσης.
 * @param bottomBase Το μήκος της κάτω βάσης.
 * @param height Το ύψος του τραπεζίου.
 * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
 */
void drawTrapezoid(float x, float y, float topBase, float bottomBase, float
height, const graphics::Brush& brush) {
    float halfHeight = height / 2;
    float halfTopBase = topBase / 2;
    float halfBottomBase = bottomBase / 2;

    graphics::drawLine(x - halfTopBase, y - halfHeight, x + halfTopBase, y -
halfHeight, brush); // επάνω γραμμή
    graphics::drawLine(x + halfTopBase, y - halfHeight, x + halfBottomBase, y +
halfHeight, brush); // δεξιά γραμμή
    graphics::drawLine(x + halfBottomBase, y + halfHeight, x - halfBottomBase, y +
halfHeight, brush); // κάτω γραμμή
    graphics::drawLine(x - halfBottomBase, y + halfHeight, x - halfTopBase, y -
halfHeight, brush); // αριστερή γραμμή
}

/**
 * Σχεδιάζει ένα τρίγωνο.
 * @param x1, y1 Συντεταγμένες πρώτης κορυφής.
 * @param x2, y2 Συντεταγμένες δεύτερης κορυφής.
 * @param x3, y3 Συντεταγμένες τρίτης κορυφής.
 * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
 */
void drawTriangle(float x1, float y1, float x2, float y2, float x3, float y3,
const graphics::Brush& brush) {
    graphics::drawLine(x1, y1, x2, y2, brush);
    graphics::drawLine(x2, y2, x3, y3, brush);
    graphics::drawLine(x3, y3, x1, y1, brush);
}

/**
 * Σχεδιάζει ένα ορθογώνιο τρίγωνο.

```

```

    * @param x1, y1 Συντεταγμένες της ορθής γωνίας.
    * @param width Το μήκος της βάσης.
    * @param height Το ύψος.
    * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
    */
    void drawRightTriangle(float x1, float y1, float width, float height, const
graphics::Brush& brush) {
        graphics::drawLine(x1, y1, x1 + width, y1, brush);
        graphics::drawLine(x1 + width, y1, x1, y1 + height, brush);
        graphics::drawLine(x1, y1 + height, x1, y1, brush);
    }

    /**
    * Σχεδιάζει ένα ισοσκελές τρίγωνο.
    * @param x, y Το κέντρο της βάσης του τριγώνου.
    * @param base Το μήκος της βάσης.
    * @param height Το ύψος.
    * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
    */
    void drawIsoscelesTriangle(float x, float y, float base, float height, const
graphics::Brush& brush) {
        graphics::drawLine(x - base / 2, y, x + base / 2, y, brush);
        graphics::drawLine(x - base / 2, y, x, y - height, brush);
        graphics::drawLine(x + base / 2, y, x, y - height, brush);
    }

    /**
    * Σχεδιάζει ένα ισόπλευρο τρίγωνο.
    * @param x, y Το κέντρο του τριγώνου.
    * @param side Το μήκος της πλευράς.
    * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
    */
    void drawEquilateralTriangle(float x, float y, float side, const graphics::Brush&
brush) {
        float height = (sqrt(3) / 2) * side;
        float halfSide = side / 2;

        graphics::drawLine(x - halfSide, y + height / 2, x + halfSide, y + height / 2,
brush);
        graphics::drawLine(x - halfSide, y + height / 2, x, y - height / 2, brush);
        graphics::drawLine(x + halfSide, y + height / 2, x, y - height / 2, brush);
    }

    /**
    * Σχεδιάζει ένα κανονικό πολύγωνο.
    * @param x H x συντεταγμένη του κέντρου.
    * @param y H y συντεταγμένη του κέντρου.
    * @param sides O αριθμός των πλευρών.
    * @param radius H απόσταση από το κέντρο έως τις κορυφές.
    * @param brush Το αντικείμενο Brush για καθορισμό του χρώματος.
    */
    void drawRegularPolygon(float x, float y, int sides, float radius, const
graphics::Brush& brush) {
        if (sides < 3) return; // Πρέπει να έχει τουλάχιστον 3 πλευρές
        float angleStep = 2 * M_PI / sides;

        for (int i = 0; i < sides; ++i) {
            float x1 = x + radius * cos(i * angleStep);
            float y1 = y + radius * sin(i * angleStep);
            float x2 = x + radius * cos((i + 1) * angleStep);
            float y2 = y + radius * sin((i + 1) * angleStep);
            graphics::drawLine(x1, y1, x2, y2, brush);
        }
    }

```

```
} // namespace GeometryShapes  
#endif // GEOMETRY_SHAPES_H
```

Πρόγραμμα χρήσης της Geometry_Shapes.h

Παρακάτω είναι ένα ολοκληρωμένο πρόγραμμα που δείχνει τη χρήση όλων των συναρτήσεων από τη βιβλιοθήκη `geometry_shapes.h`. Το πρόγραμμα δημιουργεί ένα παράθυρο και σχεδιάζει διάφορα γεωμετρικά σχήματα χρησιμοποιώντας διαφορετικά χρώματα και παραμέτρους.

```
#include "geometry_shapes.h"
#include "sgg/graphics.h"

// Μέγεθος παραθύρου
const float WINDOW_WIDTH = 800;
const float WINDOW_HEIGHT = 600;

// Συνάρτηση σχεδίασης
void draw() {
    graphics::Brush brush;

    // Καθαρισμός του παραθύρου
    brush.fill_color[0] = 0.9f;
    brush.fill_color[1] = 0.9f;
    brush.fill_color[2] = 0.9f;
    graphics::drawRect(WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2, WINDOW_WIDTH,
WINDOW_HEIGHT, brush);

    // Ρύθμιση χρώματος για τα σχήματα
    brush.fill_color[0] = 0.2f;
    brush.fill_color[1] = 0.6f;
    brush.fill_color[2] = 0.8f;

    // Σχεδίαση τετραγώνου
    GeometryShapes::drawSquare(100, 100, 50, brush);

    // Σχεδίαση παραλληλογράμμου
    GeometryShapes::drawRectangle(250, 100, 80, 50, brush);

    // Σχεδίαση ρόμβου
    GeometryShapes::drawRhombus(400, 100, 80, 50, brush);

    // Σχεδίαση ορθογώνιου παραλληλογράμμου
    GeometryShapes::drawParallelogram(550, 100, 100, 50, brush);

    // Σχεδίαση τραπεζίου
    GeometryShapes::drawTrapezoid(700, 100, 60, 100, 50, brush);

    // Αλλαγή χρώματος για τα τρίγωνα
    brush.fill_color[0] = 0.8f;
    brush.fill_color[1] = 0.3f;
    brush.fill_color[2] = 0.5f;

    // Σχεδίαση απλού τριγώνου
    GeometryShapes::drawTriangle(100, 250, 150, 250, 125, 200, brush);

    // Σχεδίαση ορθογώνιου τριγώνου
    GeometryShapes::drawRightTriangle(250, 250, 50, 30, brush);

    // Σχεδίαση ισοσκελούς τριγώνου
    GeometryShapes::drawIsoscelesTriangle(400, 250, 60, 40, brush);

    // Σχεδίαση ισοπλεύρου τριγώνου
    GeometryShapes::drawEquilateralTriangle(550, 250, 50, brush);

    // Σχεδίαση κανονικού πολυγώνου (πεντάγωνο)
```

```
    GeometryShapes::drawRegularPolygon(700, 250, 5, 30, brush);
}

// Συνάρτηση main
int main() {
    // Δημιουργία παραθύρου
    graphics::createWindow(WINDOW_WIDTH, WINDOW_HEIGHT, "Geometry Shapes Example");

    // Ορισμός της συνάρτησης σχεδίασης
    graphics::setDrawFunction(draw);

    // Εκκίνηση του κυκλώματος μηνυμάτων
    graphics::startMessageLoop();

    return 0;
}
```

Επεξήγηση

- **Ρύθμιση παραθύρου:** Το πρόγραμμα δημιουργεί ένα παράθυρο διαστάσεων 800x600.
- **Σχεδίαση σχήματος:** Η συνάρτηση `draw` περιέχει την καλεί όλες τις συναρτήσεις σχεδίασης, χρησιμοποιώντας το `Brush` για διαφορετικά χρώματα.
- **Παραδείγματα χρήσης:** Κάθε γεωμετρικό σχήμα εμφανίζεται με μια επίδειξη της αντίστοιχης συνάρτησης, ώστε να δείχνει τη μορφή και το αποτέλεσμα της σχεδίασης.

Πρόγραμμα χρήσης των Geometry_Shapes.h , brush_utils.h

Ακολουθεί πρόγραμμα που δείχνει τη χρήση των συναρτήσεων από τη βιβλιοθήκη geometry_shapes.h για τη σχεδίαση γεωμετρικών σχημάτων και τη διαχείριση του χρώματος μέσω της βιβλιοθήκης brush_utilities.h.

Πρόγραμμα

```
#include "geometry_shapes.h"
#include "brush_utils.h"
#include "sgg/graphics.h"

// Μέγεθος παραθύρου
const float WINDOW_WIDTH = 800;
const float WINDOW_HEIGHT = 600;

// Συνάρτηση για τη σχεδίαση στο παράθυρο
void draw() {
    graphics::Brush brush;

    // Δημιουργία ανοιχτού γκρι χρώματος για το φόντο
    brush_utils::setFillColor(brush, 0.9f, 0.9f, 0.9f);
    graphics::drawRect(WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2, WINDOW_WIDTH,
WINDOW_HEIGHT, brush);

    // Δημιουργία χρώματος για τα σχήματα (μπλε)
    brush_utils::setFillColor(brush, 0.2f, 0.6f, 0.8f);

    // Σχεδίαση τετραγώνου
    // Τοποθεσία: (100, 100), Πλευρά: 50
    GeometryShapes::drawSquare(100, 100, 50, brush);

    // Σχεδίαση παραλληλογράμμου
    // Τοποθεσία: (250, 100), Πλάτος: 80, Ύψος: 50
    GeometryShapes::drawRectangle(250, 100, 80, 50, brush);

    // Σχεδίαση ρόμβου
    // Τοποθεσία: (400, 100), Διαγώνιος 1: 80, Διαγώνιος 2: 50
    GeometryShapes::drawRhombus(400, 100, 80, 50, brush);

    // Σχεδίαση ορθογώνιου παραλληλογράμμου
    // Τοποθεσία: (550, 100), Πλάτος: 100, Ύψος: 50
    GeometryShapes::drawParallelogram(550, 100, 100, 50, brush);

    // Σχεδίαση τραπεζίου
    // Τοποθεσία: (700, 100), Πάνω Βάση: 60, Κάτω Βάση: 100, Ύψος: 50
    GeometryShapes::drawTrapezoid(700, 100, 60, 100, 50, brush);

    // Αλλαγή χρώματος για τα τρίγωνα (κόκκινο)
    brush_utils::setFillColor(brush, 0.8f, 0.3f, 0.5f);

    // Σχεδίαση τριγώνου
    // Κορυφές: (100, 250), (150, 250), (125, 200)
    GeometryShapes::drawTriangle(100, 250, 150, 250, 125, 200, brush);

    // Σχεδίαση ορθογώνιου τριγώνου
    // Τοποθεσία: (250, 250), Βάση: 50, Ύψος: 30
    GeometryShapes::drawRightTriangle(250, 250, 50, 30, brush);

    // Σχεδίαση ισοσκελούς τριγώνου
    // Τοποθεσία: (400, 250), Βάση: 60, Ύψος: 40
    GeometryShapes::drawIsoscelesTriangle(400, 250, 60, 40, brush);
```

```

// Σχεδίαση ισοπλεύρου τριγώνου
// Τοποθεσία: (550, 250), Πλευρά: 50
GeometryShapes::drawEquilateralTriangle(550, 250, 50, brush);

// Σχεδίαση κανονικού πενταγώνου
// Τοποθεσία: (700, 250), Ακτίνα: 30, Πλευρές: 5
GeometryShapes::drawRegularPolygon(700, 250, 5, 30, brush);
}

// Συνάρτηση main
int main() {
    // Δημιουργία παραθύρου
    graphics::createWindow(WINDOW_WIDTH, WINDOW_HEIGHT, "Example of Geometry Shapes
and Brush Utilities");

    // Ορισμός της συνάρτησης σχεδίασης
    graphics::setDrawFunction(draw);

    // Εκκίνηση του κυκλώματος μηνυμάτων
    graphics::startMessageLoop();

    return 0;
}

```

Επεξήγηση

- **Ρύθμιση χρωμάτων:** Με τη χρήση της βιβλιοθήκης `brush_utilities.h`, ορίζονται διαφορετικά χρώματα για το φόντο και τα σχήματα.
- **Σχεδίαση σχήματος:** Κάθε συνάρτηση από τη βιβλιοθήκη `geometry_shapes.h` καλείται με συγκεκριμένες παραμέτρους για τη σχεδίαση διάφορων σχημάτων.
- **Παράδειγμα χρωμάτων και γεωμετρικών σχημάτων:** Το πρόγραμμα δίνει παραδείγματα για τη χρήση διαφορετικών χρωμάτων και σχημάτων.

Βιβλιοθήκη τρισδιάστατων σχημάτων.

Βιβλιοθήκη three_d_shapes.h

Ακολουθεί ένα αρχείο header `three_d_shapes.h`, το οποίο περιέχει συναρτήσεις σχεδίασης για διάφορα τρισδιάστατα γεωμετρικά σχήματα (αναπαριστώμενα σε 2D) χρησιμοποιώντας την `sgg`.

```
#pragma once
#ifndef THREE_D_SHAPES_H
#define THREE_D_SHAPES_H

#include "sgg/graphics.h"
#include <cmath>
#define M_PI 3.14159265358979323846

namespace ThreeDShapes {

    /**
     * Σχεδιάζει έναν κύβο σε προοπτική.
     * @param x Η x-συντεταγμένη του κέντρου του κύβου.
     * @param y Η y-συντεταγμένη του κέντρου του κύβου.
     * @param side Το μήκος της πλευράς του κύβου.
     * @param brush Το αντικείμενο Brush που καθορίζει το χρώμα και το στυλ σχεδίασης.
     */
    void drawCube(float x, float y, float side, const graphics::Brush& brush) {
        float offset = side * 0.4f;
        graphics::drawRect(x, y, side, side, brush);
        graphics::drawRect(x + offset, y - offset, side, side, brush);

        graphics::drawLine(x - side / 2, y - side / 2, x + offset - side / 2, y -
offset - side / 2, brush);
        //graphics::drawLine(x + side / 2, y - side / 2, x + offset + side / 2, y -
offset - side / 2, brush);
        graphics::drawLine(x - side / 2, y + side / 2, x + offset - side / 2, y -
offset + side / 2, brush);
        graphics::drawLine(x + side / 2, y + side / 2, x + offset + side / 2, y -
offset + side / 2, brush);
    }

    /**
     * Σχεδιάζει ένα ορθογώνιο παραλληλεπίπεδο (Cuboid) σε προοπτική.
     * @param x Η x-συντεταγμένη του κέντρου του cuboid.
     * @param y Η y-συντεταγμένη του κέντρου του cuboid.
     * @param width Το πλάτος του cuboid.
     * @param height Το ύψος του cuboid.
     * @param depth Το βάθος του cuboid.
     * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
     */
    void drawCuboid(float x, float y, float width, float height, float depth, const
graphics::Brush& brush) {
        float offsetX = depth * 0.4f;
        float offsetY = depth * 0.4f;

        graphics::drawRect(x, y, width, height, brush);
        graphics::drawRect(x + offsetX, y - offsetY, width, height, brush);

        graphics::drawLine(x - width / 2, y - height / 2, x + offsetX - width / 2, y -
offsetY - height / 2, brush);
        //graphics::drawLine(x + width / 2, y - height / 2, x + offsetX + width / 2, y
- offsetY - height / 2, brush);
    }
}
```

```

        graphics::drawLine(x - width / 2, y + height / 2, x + offsetX - width / 2, y -
offsetY + height / 2, brush);
        graphics::drawLine(x + width / 2, y + height / 2, x + offsetX + width / 2, y -
offsetY + height / 2, brush);
    }

    /**
     * Σχεδιάζει έναν κώνο σε προοπτική.
     * @param x Η x-συντεταγμένη της βάσης του κώνου.
     * @param y Η y-συντεταγμένη της βάσης του κώνου.
     * @param radius Η ακτίνα της βάσης του κώνου.
     * @param height Το ύψος του κώνου.
     * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
     */
    void drawCone(float x, float y, float radius, float height, const graphics::Brush&
brush) {
        float base_offset = radius * 0.6f;
        // Σχεδίαση ελλειπτικής βάσης για προοπτική
        graphics::setScale(1.0f, 0.5f);
        graphics::drawDisk(x, y + height / 2, base_offset, brush);
        graphics::resetPose();

        // Σχεδίαση γραμμών από κορυφή προς βάση
        graphics::drawLine(x, y - height / 2, x - base_offset, y + height / 2, brush);
        graphics::drawLine(x, y - height / 2, x + base_offset, y + height / 2, brush);
    }

    /**
     * Σχεδιάζει έναν κύλινδρο σε προοπτική.
     * @param x Η x-συντεταγμένη του κέντρου του κυλίνδρου.
     * @param y Η y-συντεταγμένη του κέντρου του κυλίνδρου.
     * @param radius Η ακτίνα του κυλίνδρου.
     * @param height Το ύψος του κυλίνδρου.
     * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
     */
    void drawCylinder(float x, float y, float radius, float height, const
graphics::Brush& brush) {
        float base_offset = radius * 0.6f;

        // Σχεδίαση ελλειπτικής κορυφής και βάσης για προοπτική
        graphics::setScale(1.0f, 0.5f);
        graphics::drawDisk(x, y - height / 2, base_offset, brush); // Κορυφή
        graphics::drawDisk(x, y + height / 2, base_offset, brush); // Βάση
        graphics::resetPose();

        // Σχεδίαση γραμμών που συνδέουν κορυφή και βάση
        graphics::drawLine(x - base_offset, y - height / 2, x - base_offset, y +
height / 2, brush);
        graphics::drawLine(x + base_offset, y - height / 2, x + base_offset, y +
height / 2, brush);
    }

    /**
     * Σχεδιάζει μία σφαίρα (αναπαρίσταται ως δίσκος και ελλείψεις).
     * @param x Η x-συντεταγμένη του κέντρου της σφαίρας.
     * @param y Η y-συντεταγμένη του κέντρου της σφαίρας.
     * @param radius Η ακτίνα της σφαίρας.
     * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
     */
    void drawSphere(float x, float y, float radius, const graphics::Brush& brush) {
        graphics::drawDisk(x, y, radius, brush);
        graphics::setScale(1.0f, 0.5f);
        graphics::drawDisk(x, y, radius, brush);
        graphics::setOrientation(90);
        graphics::drawDisk(x, y, radius, brush);
        graphics::resetPose();
    }

```

```

}

/**
 * Σχεδιάζει μία πυραμίδα σε προοπτική.
 * @param x H x-συντεταγμένη του κέντρου της βάσης της πυραμίδας.
 * @param y H y-συντεταγμένη του κέντρου της βάσης της πυραμίδας.
 * @param base_side H πλευρά της τετράγωνης βάσης της πυραμίδας.
 * @param height Το ύψος της πυραμίδας.
 * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
 */
void drawPyramid(float x, float y, float base_side, float height, const
graphics::Brush& brush) {
    // Ορισμός της κλίσης για καλύτερη προοπτική στην βάση της πυραμίδας
    float offsetX = base_side * 0.5f; // μισό της πλευράς βάσης για συμμετρία
    float offsetY = base_side * 0.3f; // μικρότερος συντελεστής για "συμπίεση"
    της προοπτικής

    // Σχεδίαση της βάσης της πυραμίδας ως παραλληλόγραμμο με προοπτική
    graphics::drawLine(x - offsetX, y - offsetY, x + offsetX, y - offsetY, brush);
    // πάνω πλευρά
    graphics::drawLine(x - offsetX, y + offsetY, x + offsetX, y + offsetY, brush);
    // κάτω πλευρά
    graphics::drawLine(x - offsetX, y - offsetY, x - offsetX, y + offsetY, brush);
    // αριστερή πλευρά
    graphics::drawLine(x + offsetX, y - offsetY, x + offsetX, y + offsetY, brush);
    // δεξιά πλευρά

    // Συντεταγμένες κορυφής της πυραμίδας
    float peakX = x;
    float peakY = y - height;

    // Σχεδίαση γραμμών από την κορυφή της πυραμίδας στις γωνίες της βάσης για
    καλύτερη προοπτική
    graphics::drawLine(peakX, peakY, x - offsetX, y - offsetY, brush); // από
    κορυφή σε αριστερή πάνω γωνία
    graphics::drawLine(peakX, peakY, x + offsetX, y - offsetY, brush); // από
    κορυφή σε δεξιά πάνω γωνία
    graphics::drawLine(peakX, peakY, x - offsetX, y + offsetY, brush); // από
    κορυφή σε αριστερή κάτω γωνία
    graphics::drawLine(peakX, peakY, x + offsetX, y + offsetY, brush); // από
    κορυφή σε δεξιά κάτω γωνία
}

/**
 * Σχεδιάζει ένα πρίσμα με κανονική βάση σε προοπτική.
 * @param x H x-συντεταγμένη του κέντρου του πρίσματος.
 * @param y H y-συντεταγμένη του κέντρου του πρίσματος.
 * @param sides O αριθμός των πλευρών της βάσης του πρίσματος.
 * @param radius H ακτίνα της βάσης του πρίσματος.
 * @param height Το ύψος του πρίσματος.
 * @param brush Το αντικείμενο Brush για το στυλ σχεδίασης.
 */
void drawPrism(float x, float y, int sides, float radius, float height, const
graphics::Brush& brush) {
    std::vector<float> px(sides), py(sides);
    for (int i = 0; i < sides; i++) {
        float angle = 2 * M_PI * i / sides;
        px[i] = x + radius * cos(angle);
        py[i] = y + radius * sin(angle);
    }

    for (int i = 0; i < sides; i++) {
        graphics::drawLine(px[i], py[i], px[(i + 1) % sides], py[(i + 1) % sides],
brush);
    }
}

```

```

        graphics::drawLine(px[i], py[i], px[i], py[i] - height, brush);
    }

    for (int i = 0; i < sides; i++) {
        graphics::drawLine(px[i], py[i] - height, px[(i + 1) % sides], py[(i + 1)
% sides] - height, brush);
    }
}

} // namespace ThreeDShapes

#endif // THREE_D_SHAPES_H

```

Επεξήγηση

- **drawCube:** Σχεδιάζει έναν κύβο με δύο τετράγωνα και τις συνδετικές γραμμές για προοπτική.
- **drawCuboid:** Παρόμοιο με τον κύβο, αλλά υποστηρίζει διαφορετικό πλάτος, ύψος και βάθος.
- **drawCone:** Σχεδιάζει τη βάση του κώνου ως δίσκο και προσθέτει δύο γραμμές που συγκλίνουν στην κορυφή.
- **drawCylinder:** Σχεδιάζει έναν κύλινδρο με δίσκους για την κορυφή και τη βάση και τις αντίστοιχες γραμμές στα άκρα.
- **drawSphere:** Σχεδιάζει τη σφαίρα ως δίσκο με δύο ελλείψεις για να δοθεί η αίσθηση της τρισδιάστατης προβολής.
- **drawPyramid:** Σχεδιάζει μία πυραμίδα με βάση και γραμμές που οδηγούν στην κορυφή.
- **drawPrism:** Σχεδιάζει ένα πρίσμα με κανονική βάση και καθορισμένο ύψος.

Με αυτό το header file, μπορείτε να σχεδιάσετε βασικά τρισδιάστατα σχήματα σε ένα 2D παράθυρο, δίνοντας προοπτική και διαχείριση των διαστάσεων και του βάθους.

Πρόγραμμα χρήσης της "three_d_shapes.h".

Ακολουθεί πρόγραμμα που χρησιμοποιεί τη βιβλιοθήκη three_d_shapes.h για να σχεδιάσει τα τρισδιάστατα σχήματα σε 2D, χρησιμοποιώντας προοπτική.

```
#include "three_d_shapes.h" // Συμπερίληψη της βιβλιοθήκης για τα τρισδιάστατα σχήματα
#include "sgg/graphics.h"   // Συμπερίληψη της βιβλιοθήκης γραφικών sgg
#include <cmath>

// Διαστάσεις παραθύρου
const float WINDOW_WIDTH = 800;
const float WINDOW_HEIGHT = 600;

void draw() {
    graphics::Brush brush;

    // Ορισμός χρώματος για τον κύβο
    brush.fill_color[0] = 0.2f;
    brush.fill_color[1] = 0.6f;
    brush.fill_color[2] = 0.8f;
    ThreeDShapes::drawCube(150, 150, 100, brush); // Σχεδίαση κύβου

    // Ορισμός χρώματος για το παραλληλεπίπεδο
    brush.fill_color[0] = 0.8f;
    brush.fill_color[1] = 0.5f;
    brush.fill_color[2] = 0.2f;
    ThreeDShapes::drawCuboid(400, 150, 120, 80, 60, brush); // Σχεδίαση
    παραλληλεπιπέδου

    // Ορισμός χρώματος για τον κώνο
    brush.fill_color[0] = 1.0f;
    brush.fill_color[1] = 0.0f;
    brush.fill_color[2] = 0.0f;
    ThreeDShapes::drawCone(650, 150, 50, 100, brush); // Σχεδίαση κώνου

    // Ορισμός χρώματος για τον κύλινδρο
    brush.fill_color[0] = 0.3f;
    brush.fill_color[1] = 1.0f;
    brush.fill_color[2] = 0.3f;
    ThreeDShapes::drawCylinder(150, 400, 40, 120, brush); // Σχεδίαση κυλίνδρου

    // Ορισμός χρώματος για τη σφαίρα
    brush.fill_color[0] = 1.0f;
    brush.fill_color[1] = 1.0f;
    brush.fill_color[2] = 0.0f;
    ThreeDShapes::drawSphere(400, 300, 50, brush); // Σχεδίαση σφαίρας

    // Ορισμός χρώματος για την πυραμίδα
    brush.fill_color[0] = 0.6f;
    brush.fill_color[1] = 0.3f;
    brush.fill_color[2] = 0.9f;
    ThreeDShapes::drawPyramid(650, 400, 80, 100, brush); // Σχεδίαση πυραμίδας

    // Ορισμός χρώματος για το πρίσμα
    brush.fill_color[0] = 1.0f;
    brush.fill_color[1] = 0.5f;
    brush.fill_color[2] = 0.0f;
    ThreeDShapes::drawPrism(400, 550, 6, 40, 80, brush); // Σχεδίαση εξαγωνικού
    πρίσματος
}

int main() {
    // Δημιουργία παραθύρου για τη σχεδίαση
```

```

graphics::createWindow(WINDOW_WIDTH, WINDOW_HEIGHT, "Three-Dimensional Shapes");

// Ορισμός της συνάρτησης σχεδίασης
graphics::setDrawFunction(draw);

// Εκκίνηση του κύκλου μηνυμάτων
graphics::startMessageLoop();

return 0;
}

```

Επεξήγηση Προγράμματος

- Αρχικοποίηση Παραθύρου:** Δημιουργείται ένα παράθυρο με διαστάσεις 800x600 για να εμφανιστούν τα σχήματα.
- Ορισμός Σχήματος:** Κάθε σχήμα σχεδιάζεται στο παράθυρο, χρησιμοποιώντας τις συναρτήσεις της βιβλιοθήκης ThreeDShapes. Για κάθε σχήμα:
 - Ρυθμίζουμε το χρώμα του `brush` για κάθε σχήμα (π.χ., κύβος, κώνος, κλπ.).
 - Καλούμε την αντίστοιχη συνάρτηση της βιβλιοθήκης (π.χ., `drawCube`, `drawCylinder`, κλπ.) με τις κατάλληλες παραμέτρους, όπως το κέντρο, η διάμετρος, το ύψος, κ.λπ.
- Συναρτήσεις Σχεδίασης:** Οι συναρτήσεις σχεδίασης καλούνται από τη `draw`, η οποία εκτελείται κάθε φορά που ανανεώνεται η οθόνη.
- Κύκλος Μηνυμάτων:** Το `graphics::startMessageLoop()` διαχειρίζεται τη συνεχή λειτουργία του προγράμματος μέχρι ο χρήστης να κλείσει το παράθυρο.

Παρατηρήσεις

- Το πρόγραμμα χρησιμοποιεί διαφορετικά χρώματα για κάθε σχήμα ώστε να φαίνονται καθαρά οι διαφορές τους.
- Οι συντεταγμένες κάθε σχήματος είναι ρυθμισμένες ώστε να κατανέμονται στο παράθυρο ομοιόμορφα.
- Η `drawPrism` χρησιμοποιεί εξαγωνική βάση με 6 πλευρές, αλλά μπορεί να προσαρμοστεί για οποιοδήποτε πολυγωνικό πρίσμα.

Πρόγραμμα χρήσης της "three_d_shapes.h", "brush_utils.h"

Ακολουθεί η τροποποιημένη έκδοση του προγράμματος που χρησιμοποιεί τη βιβλιοθήκη `brush_utils.h` για τον χειρισμό του χρώματος:

```
#include "three_d_shapes.h" // Συμπερίληψη της βιβλιοθήκης για τα τρισδιάστατα σχήματα
#include "brush_utils.h"    // Συμπερίληψη της βιβλιοθήκης για χειρισμό του χρώματος
#include "sgg/graphics.h"  // Συμπερίληψη της βιβλιοθήκης γραφικών
#include <cmath>

// Διαστάσεις παραθύρου
const float WINDOW_WIDTH = 800;
const float WINDOW_HEIGHT = 600;

void draw() {
    graphics::Brush brush;

    // Ορισμός χρώματος για τον κύβο με τη χρήση της brush_utils
    brush_utils::setFillColor(brush, 0.2f, 0.6f, 0.8f);
    ThreeDShapes::drawCube(150, 150, 100, brush); // Σχεδίαση κύβου

    // Ορισμός χρώματος για το παραλληλεπίπεδο με τη χρήση της brush_utils
    brush_utils::setFillColor(brush, 0.8f, 0.5f, 0.2f);
    ThreeDShapes::drawCuboid(400, 150, 120, 80, 60, brush); // Σχεδίαση
    παραλληλεπιπέδου

    // Ορισμός χρώματος για τον κώνο
    brush_utils::setFillColor(brush, 1.0f, 0.0f, 0.0f);
    ThreeDShapes::drawCone(650, 150, 50, 100, brush); // Σχεδίαση κώνου

    // Ορισμός χρώματος για τον κύλινδρο
    brush_utils::setFillColor(brush, 0.3f, 1.0f, 0.3f);
    ThreeDShapes::drawCylinder(150, 400, 40, 120, brush); // Σχεδίαση κυλίνδρου

    // Ορισμός χρώματος για τη σφαίρα
    brush_utils::setFillColor(brush, 1.0f, 1.0f, 0.0f);
    ThreeDShapes::drawSphere(400, 300, 50, brush); // Σχεδίαση σφαίρας

    // Ορισμός χρώματος για την πυραμίδα
    brush_utils::setFillColor(brush, 0.6f, 0.3f, 0.9f);
    ThreeDShapes::drawPyramid(650, 400, 80, 100, brush); // Σχεδίαση πυραμίδας

    // Ορισμός χρώματος για το πρίσμα
    brush_utils::setFillColor(brush, 1.0f, 0.5f, 0.0f);
    ThreeDShapes::drawPrism(400, 550, 6, 40, 80, brush); // Σχεδίαση εξαγωνικού
    πρίσματος
}

int main() {
    // Δημιουργία παραθύρου για τη σχεδίαση
    graphics::createWindow(WINDOW_WIDTH, WINDOW_HEIGHT, "Three-Dimensional Shapes");

    // Ορισμός της συνάρτησης σχεδίασης
    graphics::setDrawFunction(draw);

    // Εκκίνηση του κύκλου μηνυμάτων
    graphics::startMessageLoop();

    return 0;
}
```

Επεξηγήσεις:

1. **Χρήση της `brush_utils`:** Η `brush_utils::setFillColor` χρησιμοποιείται για τον καθορισμό του χρώματος του `brush`, κάνοντάς το πιο ευανάγνωστο και επεκτάσιμο.
2. **Αναλυτικά Σχόλια:** Ο κώδικας περιέχει σχόλια στα ελληνικά που εξηγούν τη λειτουργία του κάθε τμήματος, από τον καθορισμό του χρώματος μέχρι τη σχεδίαση του κάθε σχήματος.