1 Struktury danych

1.1 Deque

Dwukierunkowa kolejka została użyta do przechowywania elementów krzywej w postaci segmentów. Zaimplementowana w postaci kontenera biblioteki standardowej C++ std::deque.

1.2 Color

```
struct Color
{
    Color(int _R, int _G, int _B) : R(_R), G(_G), B(_B) {}
    Color next();
    Color &operator+=(int i);

    int R;
    int G;
    int B;
private:
    enum cycle
    {
        Ru,
        Gu,
        Rd,
        Bu,
        Gd,
        Bd,
    };
    cycle current_cycle = Ru;
    int m_step = 1;
};
```

Struktura przechowująca kolor w postaci kodu RGB.
Metoda `next()` pozwala wygenerować następny kolor według schematu opisanego w sekcji Algorytmy.
`operator+=(int i)` służy do zwiększenia jasności koloru poprzez dodanie `i` do każdego ze składników RGB.

1.3 Point

```
struct Point
{
    Point() = default;
    Point(double x, double y, double z);
    Point as_spherical();

    double x = 0;
    double y = 0;
    double z = 0;
};
```

Struktura przechowuje położenie punktu w postaci trzech współrzędnych w układzie kartezjańskim lub sferycznym.
Metoda `as_spherical()` pozwala na interpretację punktu zapisanego we współrzędnych sferycznych w układzie kartezjańskim (co jest konieczne aby poprawnie wyświetlić go na ekranie).

1.4 Segment

```
struct Segment
{
    Segment(Point _begin, Point _end, Color _color) :
        begin(_begin), end(_end), color(_color) {}

    Point begin;
    Point end;
    Color color;
};
```

Struktura przechowująca informacje dotyczące pojedynczego odcinka w postaci: punkt początkowy, punkt końcowy, kolor odcinka.

1.5 Vector4

```
struct Vector4
{
    friend Vector4 operator*(const Vector4&, double);

    Vector4();
    Vector4 operator-(const Vector4&);

    void set(double d1, double d2, double d3);
    double get_x();
    double get_y();
    double get_z();

    double data[4];
};
```

Struktura przechowująca wektor w postaci [x, y, z, t]. Podczas inicjalizacji t jest ustawiane na 1.
`operator*` pozwala wykonać mnożenie wektora przez skalar.

1.6 Matrix4

```
struct Matrix4
{
    friend Vector4 operator*(const Matrix4, const Vector4);

    Matrix4();
    Matrix4 operator*(const Matrix4);

    double data[4][4];
};
```

Struktura zawierająca macierz 4x4.
Pozwala na operacje mnożenia wektorowego macierz $\times$ wektor i macierz $\times$ macierz.