

課題の目標

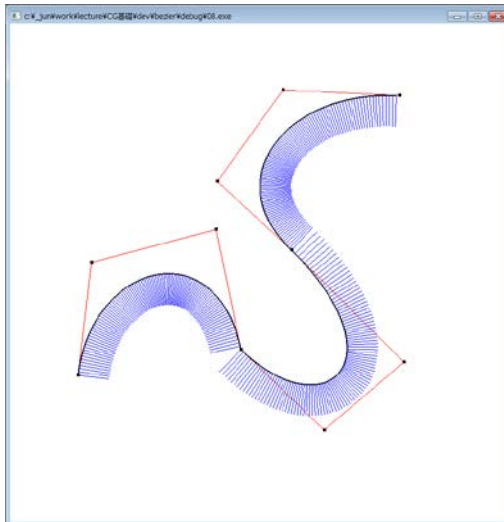
- ・パラメトリック曲線の1つであるベジエ曲線の仕組みを理解する
- ・制御点を入力することで、ベジエ曲線を描画するアプリケーションの開発を行う
- ・C++言語の便利な機能の一つである STL の **vector** (可変長配列) を使えるようになる

課題の内容

- (1) サンプルコード `stl_vector_example.cpp` をコンパイル・実行し、C++言語で2次元ベクトルを扱ったりサイズを変更可能な配列 **vector** を扱ったりする方法を確認しなさい (2次元ベクトルを扱う **Vector2d** クラスと、可変長配列を扱う **vector** は名前が似ているが全く異なるものです)。その後、コメント文に記された課題の内容に従ってプログラムコードを追加しなさい。
- (2) サンプルコード `kadai05_sample.cpp` に対して、下記の条件を満たすように `display()` 関数を完成させなさい。プログラムができたなら、なるべく滑らかに $\infty$ 記号を描いてみなさい。また、なるべく綺麗な円を描いてみなさい。

(条件)

1. マウスの左クリックで制御点を追加、右クリックで削減する (サンプルコードで実現済み)。
2. 入力された制御点を使って、3次の **Bezier** 曲線を描画する。下図のように制御点の数に応じて、複数のベジエ曲線が連結するようにすること。



つまり、制御点を  $P_0, P_1, P_2, P_3 \dots$  とすると、1 つめのセグメントは  $P_3$  が入力されたタイミングで  $P_0 \sim P_3$  の 4 点を使って描画され、2 つめのセグメントは  $P_6$  が入力されたタイミングで  $P_3 \sim P_6$  の 4 点を使って描画される (制御点  $P_3$  を 1 つめのセグメントと共有する)。3 つめの制御点は  $P_9$  が入力されたタイミングで  $P_6 \sim P_9$  の 4 点を使って描画される (制御点  $P_6$  を 2 つめのセグメントと共有する)。以下同様。セグメント数に制限は設けない。

3. 上図のように **Bezier** 曲線の法線 (接線を時計回りに 90 度回転させたもの) を描画する (長さは任意でよい)。

(発展課題：オプション)

一般化された  $N$  次の **Bezier** 曲線を描画する（法線は描画しなくてよい）。次数は **3** を初期状態とし、キーボードで上げ下げできるようにする。