宇宙機での利用に向けたデバイス開発に係る可視化
---単一磁束量子回路の例---

三浦 昭[*1]　松崎 恵一[*1]　石田 貴行[*1]
田中 雅光[*2]　井上 弘士[*3]

*1 宇宙航空研究開発機構
*2 名古屋大学
*3 九州大学

宇宙科学データの可視化

- テレメトリデータ等の可視化
  - 観測データ、軌道データ等

- 直近の計画に対して
  - 模擬データの生成・可視化
  - 例: リュウゴイド

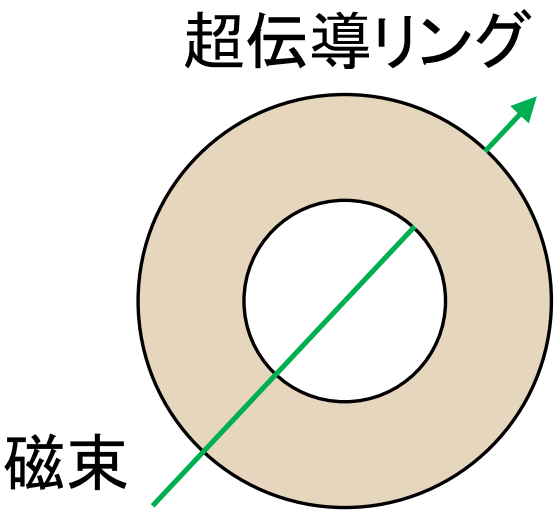
- 将来計画や構想に対して
  ←いまここ

単一磁束量子回路

- 超伝導デバイスを用いた演算回路を構想
  - これからの（研究途上の）デバイス
  - 宇宙機への応用の構想
- CMOS由来の論理回路とは異なる動作原理
- 可視化により，その動作原理の理解を深める

単一磁束量子回路

- **超伝導リング**



超伝導リングの中を通る磁束が
量子化される.

$$\Phi_0 = 2.067833\times10^{-15}[\mathrm{Wb}]$$

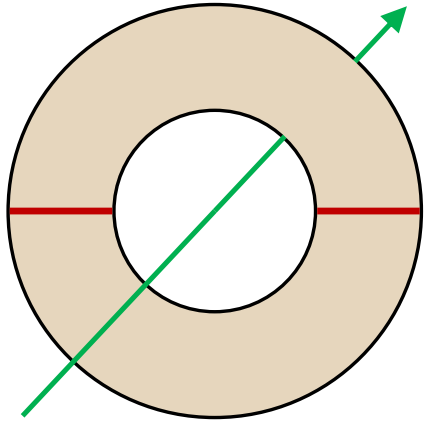
の整数倍

位相差が
2πの整数倍

単一磁束量子回路

- ジョセフソン接合を含むリング

超伝導リング



ジョセフソン接合→
（超伝導ではない
薄い結合部分）

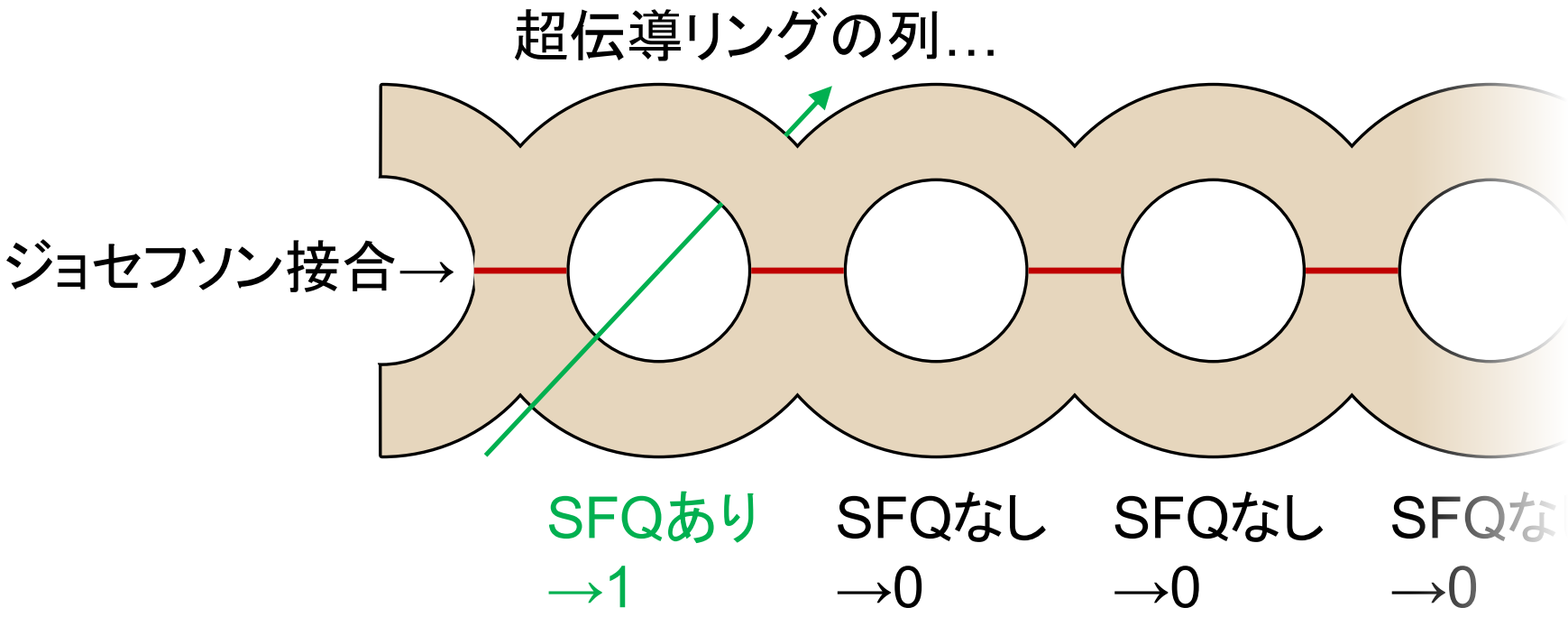
ジョセフソン接合込みで
位相差が
2πの整数倍

超伝導リングの中を通るfluxoid
（磁束もどき）が量子化される.

磁束量子の有無で1,0を表す
⇔単一磁束量子
SFQ(Single Flux Quantum)

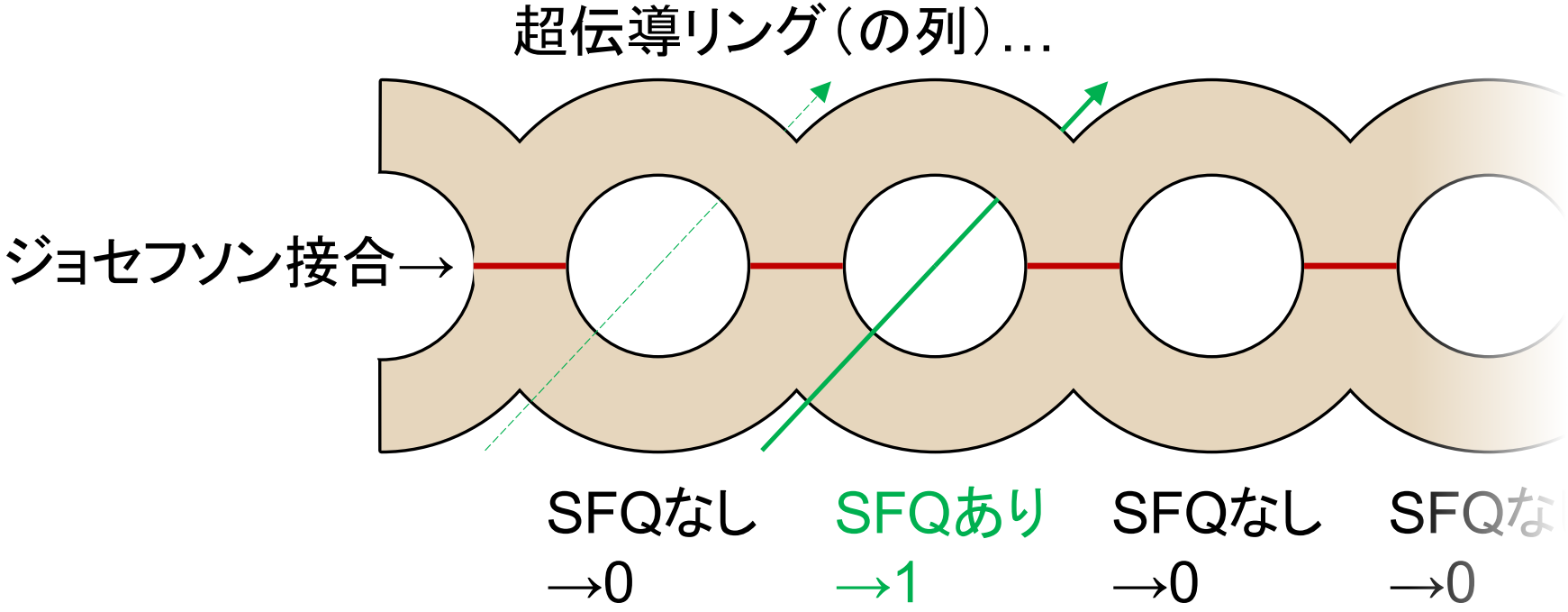
単一磁束量子回路

- ジョセフソン伝送路（JTL: Josephson Transmission Line)



単一磁束量子回路

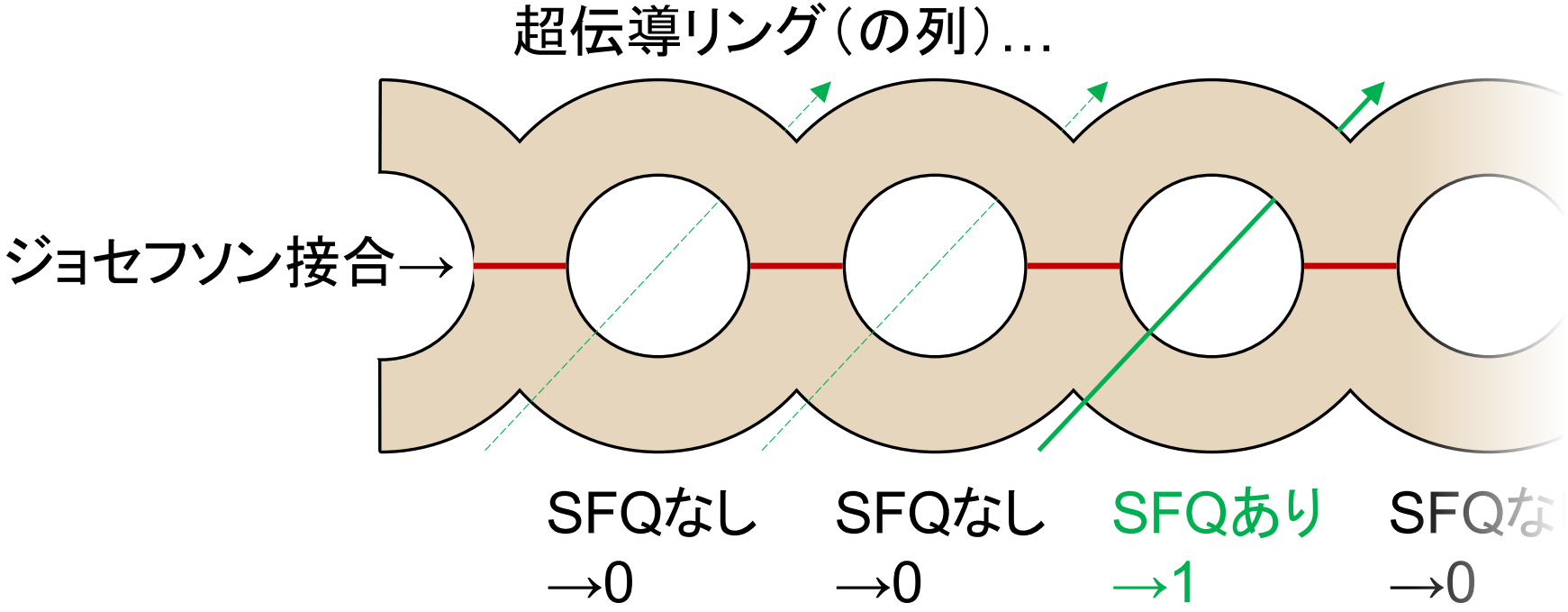
- **ジョセフソン伝送路（JTL: Josephson Transmission Line)**



適切に回路を構成すると，超高速でSFQが伝搬する.
e.g., パルス電圧: ～1 [mV], パルス幅: [ps]（⇔サブTHz）のオーダ

単一磁束量子回路

- **ジョセフソン伝送路（JTL: Josephson Transmission Line)**



適切に回路を構成すると，超高速でSFQが伝搬する.
e.g., パルス電圧: ～1 [mV], パルス幅: [ps]（⇔サブTHz）のオーダ
超高速・超低消費電力?!

しかしよくわからない…

わかりやすい表現は…

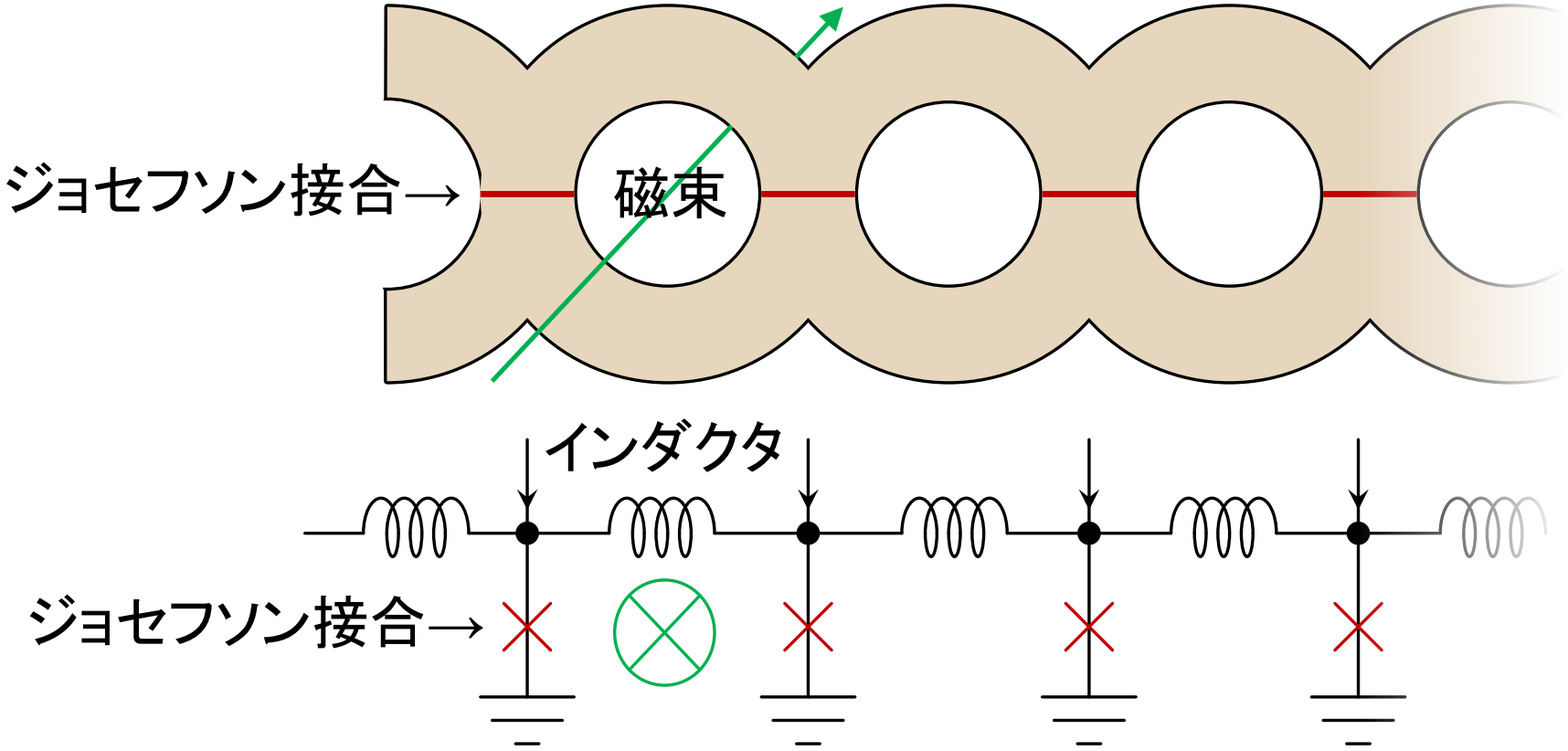
これまでに試みられてきた ジョセフソン接合のモデル

- 等価回路
- 振り子モデル
- 洗濯板モデル
  - 粘性液体中の傾いた洗濯板の上を運動する質点

- 見た目にわかりやすいモデルで表現したい

等価回路

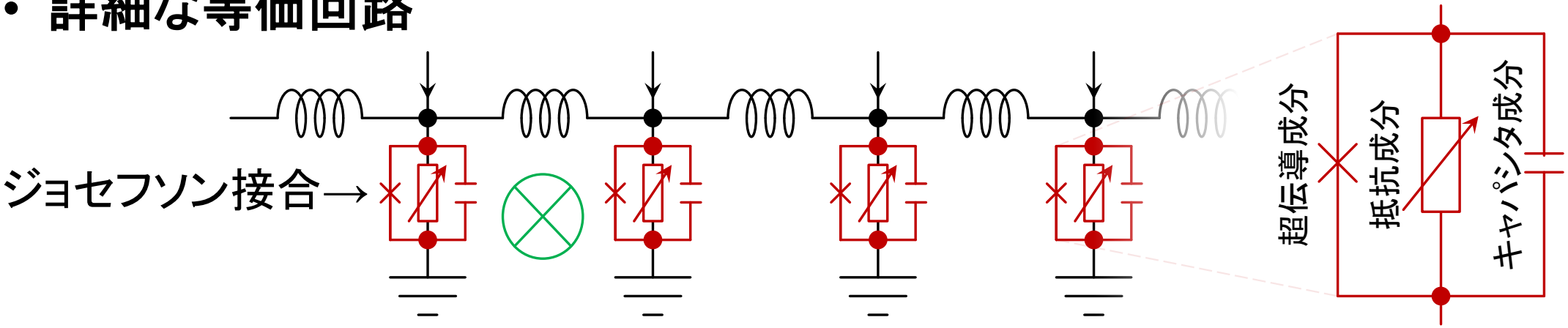
- ジョセフソン伝送路を電気回路として読み解く



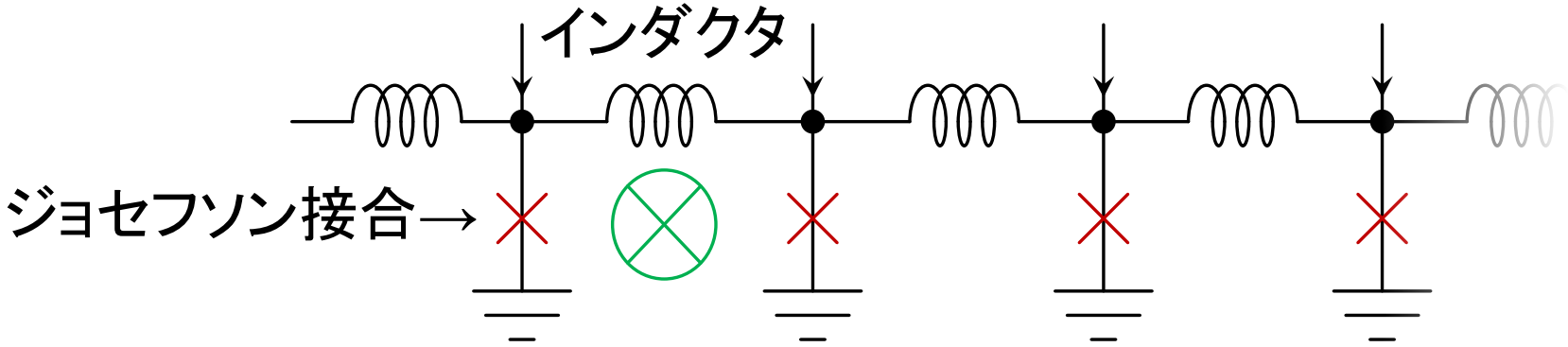
K. K. Likharev and V. K. Semenov, “RSFQ Logic/Memory Family: A New Josephson-Junction Technology for Sub- Terahertz-Clock-Frequency Digital Systems”, IEEE Trans. Appl. Supercond. 1, pp.3-28, 1991.

等価回路

- 詳細な等価回路

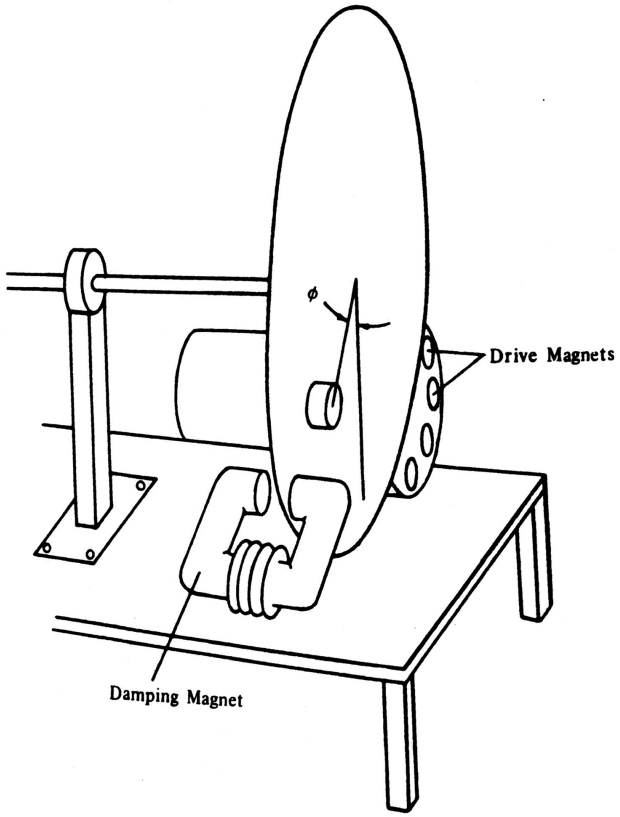


D. E. McCumber, “Effect of ac Impedance on dc Voltage- Current Characteristics of Superconductor Weak-Link Junctions,” J. Appl. Phys., vol. 39, no. 7, pp.3113–3118, 1968.



振り子モデル

- ジョセフソン接合を振り子に喩える
- 構成要素の対比
  - 位相差→回転角
  - 電圧　→角速度
  - ジョセフソン接合部分
    - ・超電導成分
    - ・抵抗成分→減衰定数$^{-1}$
    - ・キャパシタ成分→慣性モーメント
  - 電源電流→トルク



T. Van Duzer and C. W. Turner, “Principles of Superconductive Devices and Circuits”, p. 202, Feb. 1982

どう可視化するか?

Webベースで構築

- WebGL
  - 業界標準の3D CG環境
  - 様々なブラウザでサポートされている
  - 今回の用途としては実用レベルの3D CG
- JavaScript→WWWサーバ
  - コンパイル、パッケージング不要
  - 即座に可視化
  - インタラクティブ
  - リモートでも迅速に共有できる
- データ量は?
  - 軽量データ
  - モバイル環境でも共有可

今回用いたライブラリ（API）

- 3D CG
  - xeogl
    - WebGLベースの 3D CGライブラリ
    - 3D オブジェクト作成の労力を軽減
- 各種パラメータ表示・入力
  - dat.GUI
    - 数値，バー，チェックボックス
    - メニューの展開・折り畳み
- ファイルI/O
  - jQuery

→コーディングに費やす時間を削減

- どう可視化するかに注力

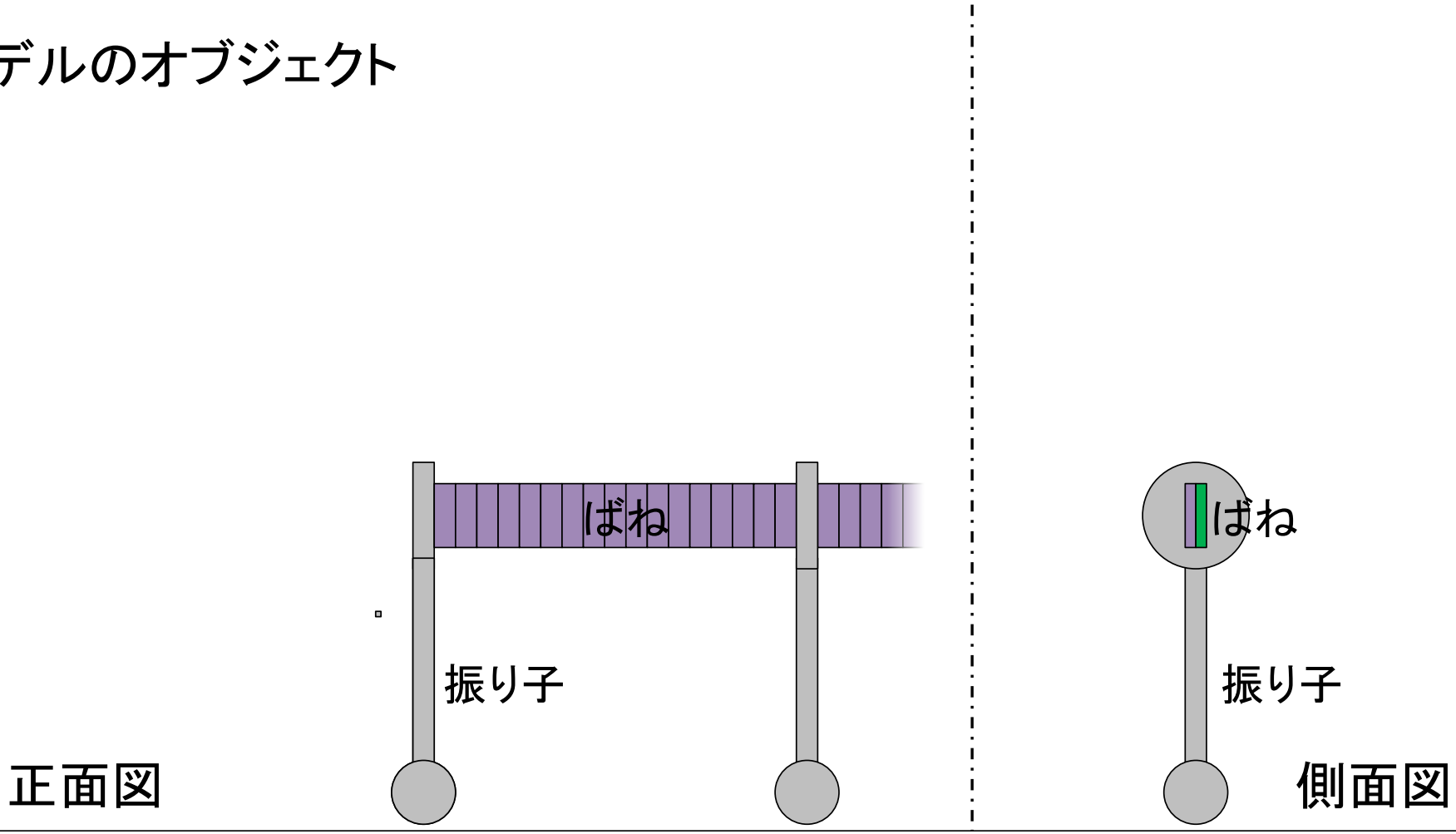
このたびの可視化対象

- 振り子モデル
  - ジョセフソン接合部位の位相差（＋電圧）に注目
- 振り子同士の連結（等価回路のインダクタ部分）
  - 前例
    - コイルばね
    - ゴム紐
  - 今回
    - 複数枚の板でねじれを表現
- 等価回路
  - 上記以外の要素を可視化
  - 電流，電圧の時間変化

可視化要素の配置

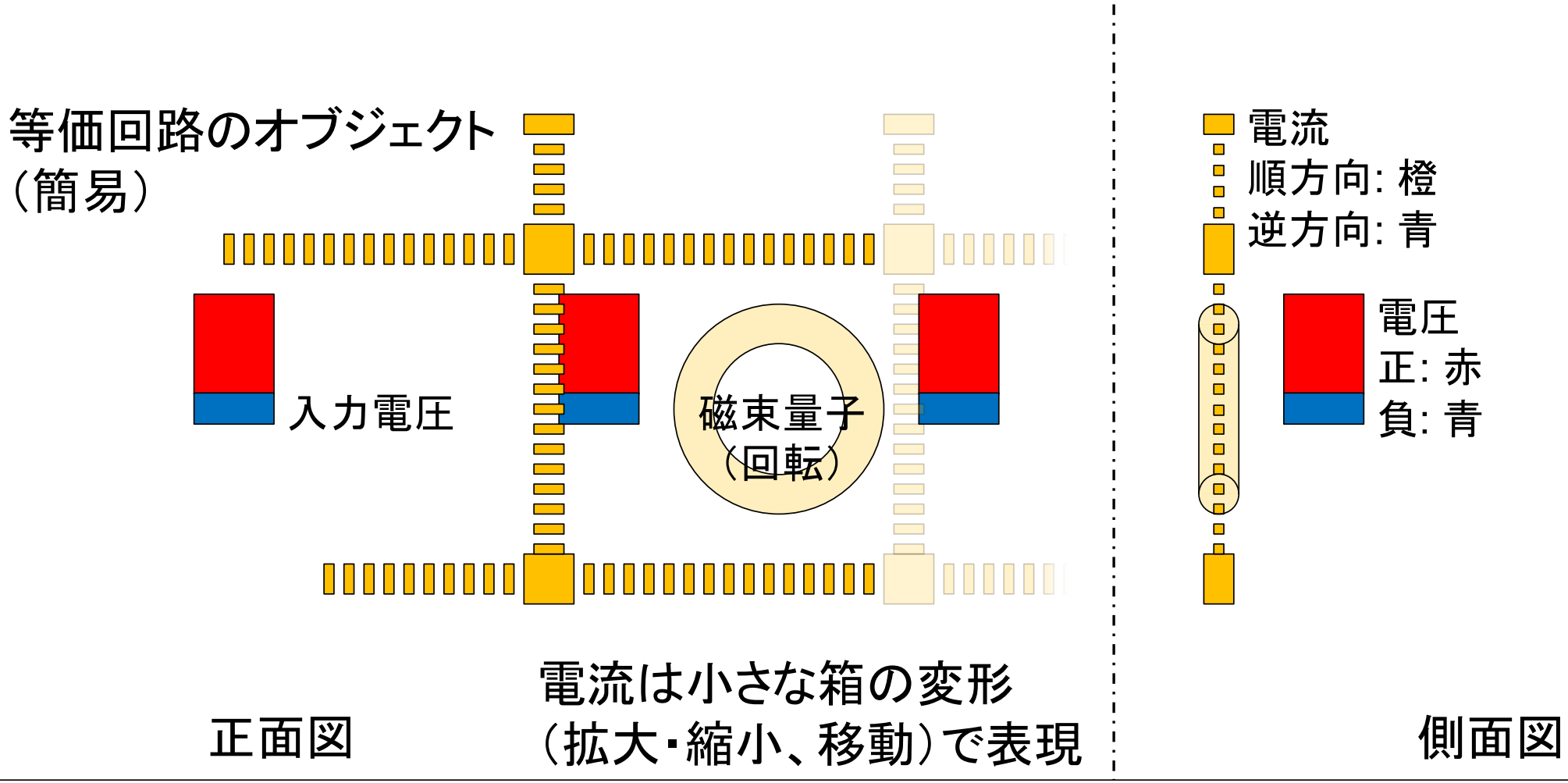
- **xeoglベースでオブジェクトを配置する**

振り子モデルのオブジェクト



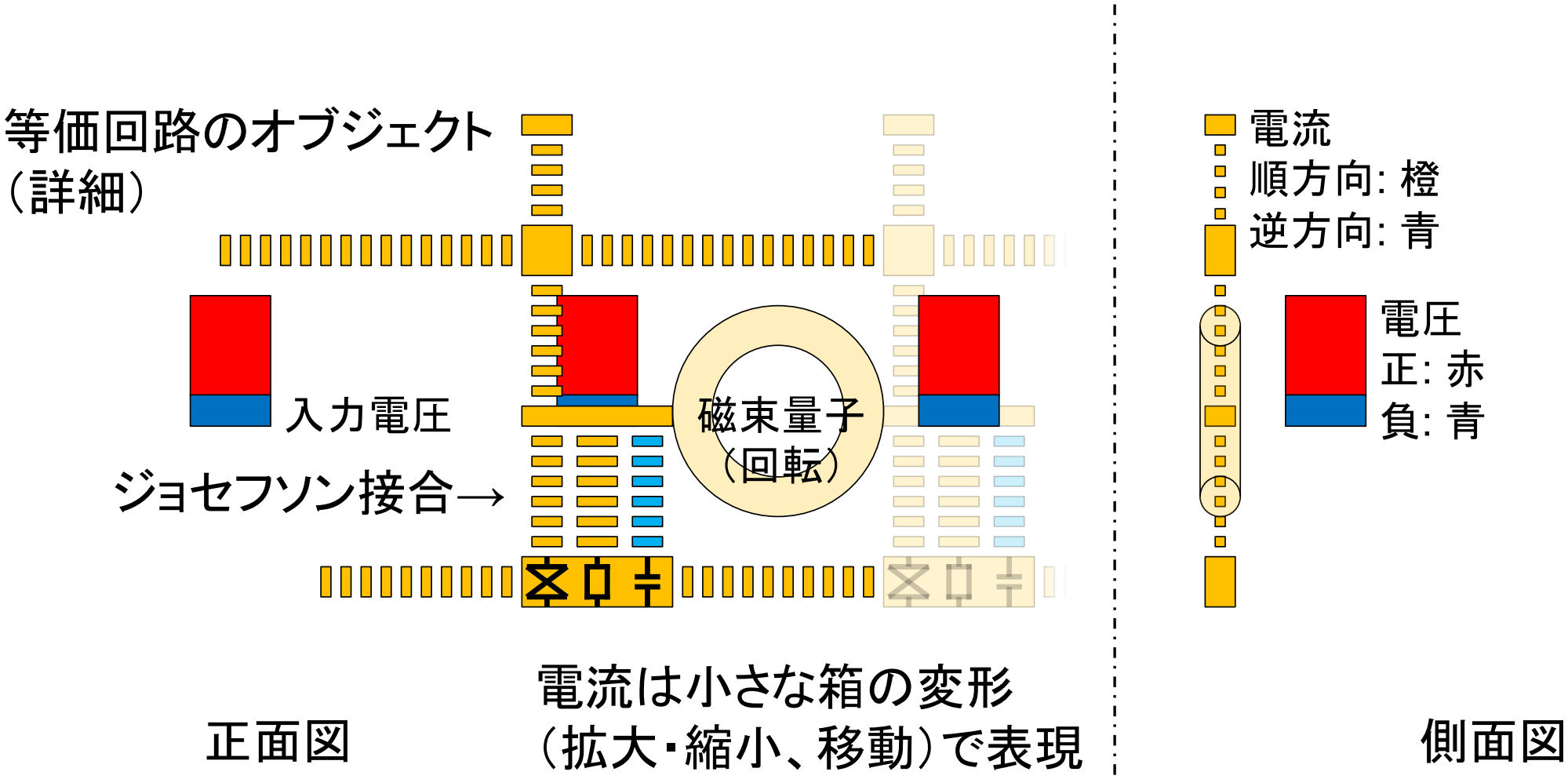
可視化要素の配置

- xeoglベースで3Dオブジェクトを配置する



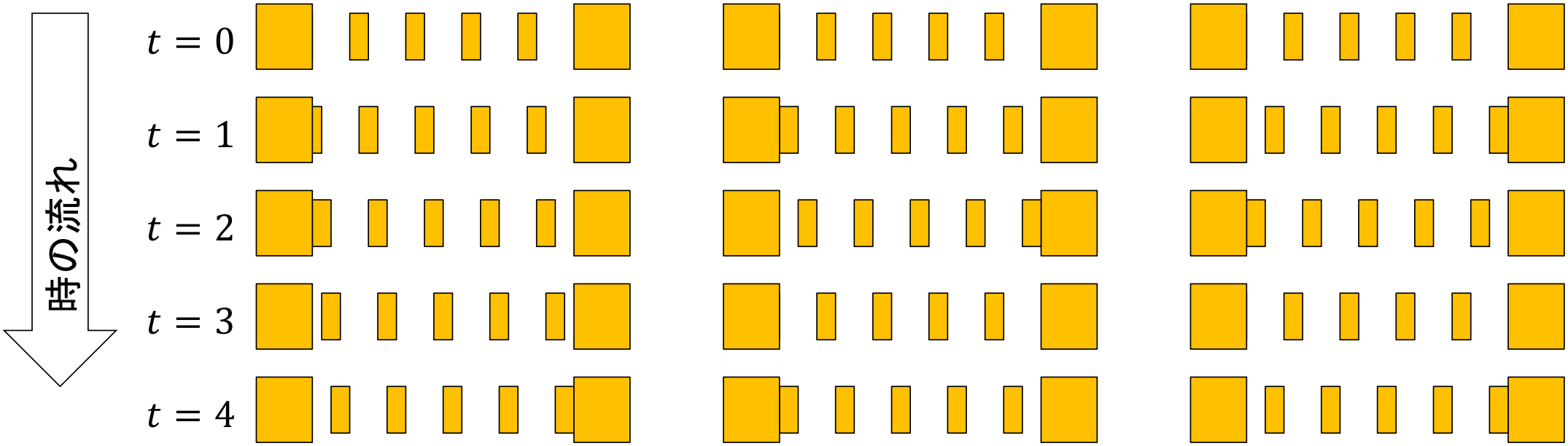
可視化要素の配置

- xeoglベースで3Dオブジェクトを配置する



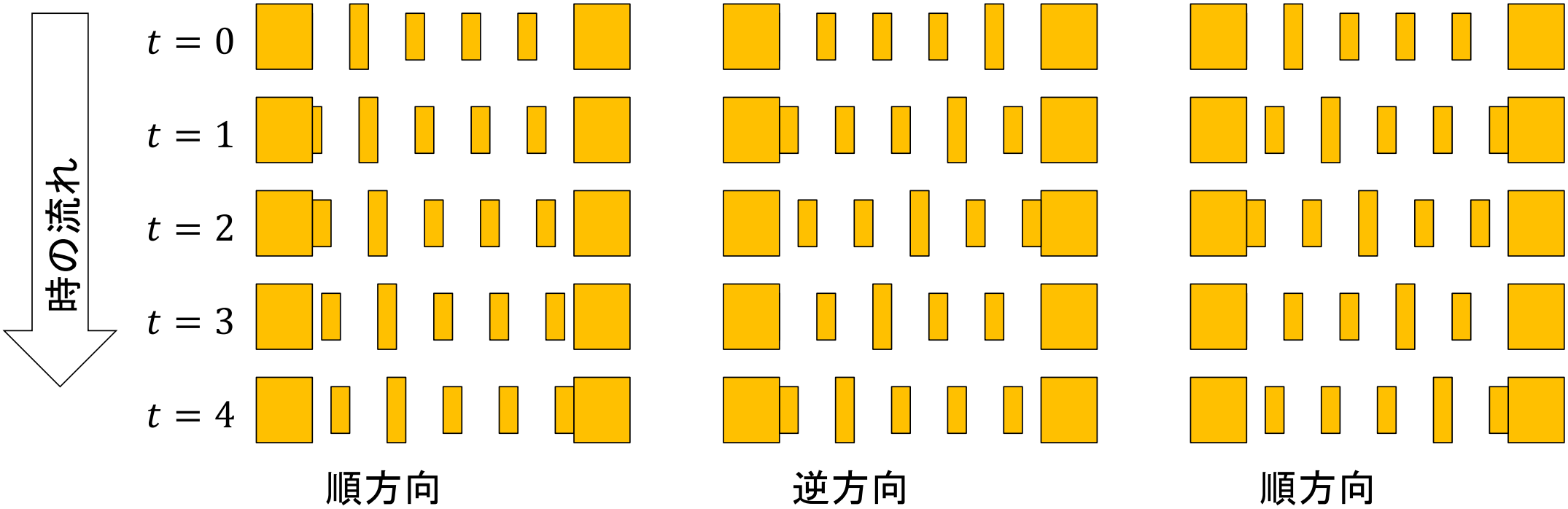
電流の表現（エイリアシング対策）

- どちら向きに動いている?
  - 今回のモデルは、電流を箱の連なり（■　■　■…）で表現ている
  - 大きな電流（箱の高速移動）に対してはエイリアシングが発生する



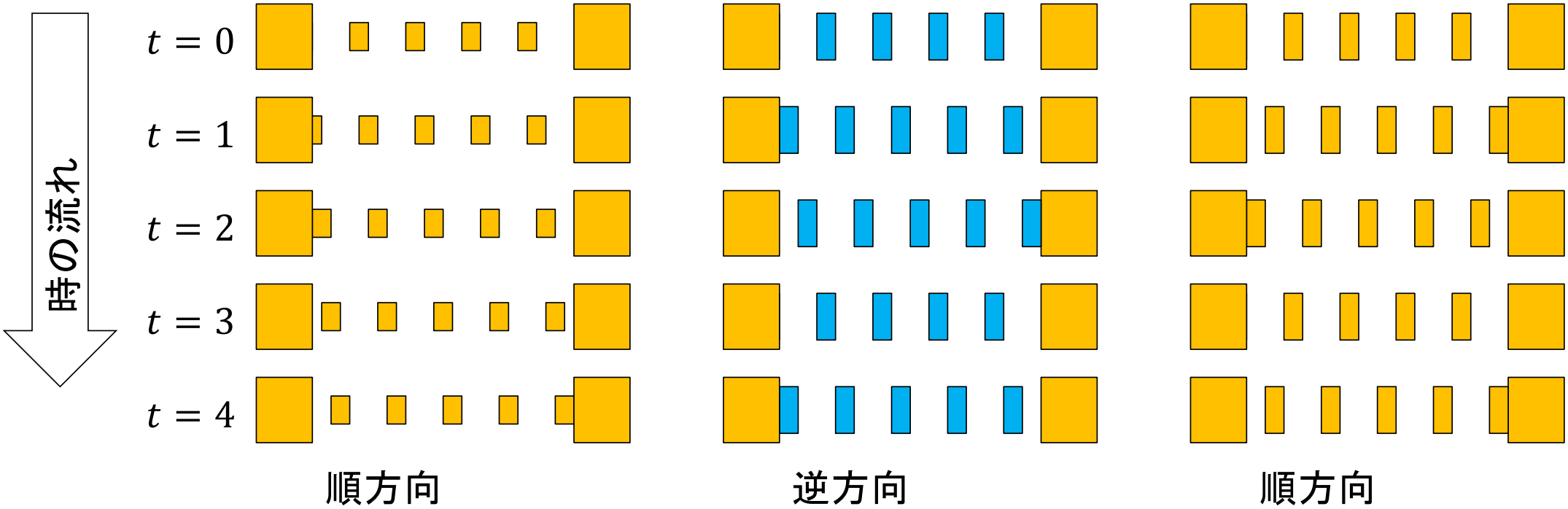
電流の表現（エイリアシング対策）

- どちら向きに動いている?
  - 目印を入れる



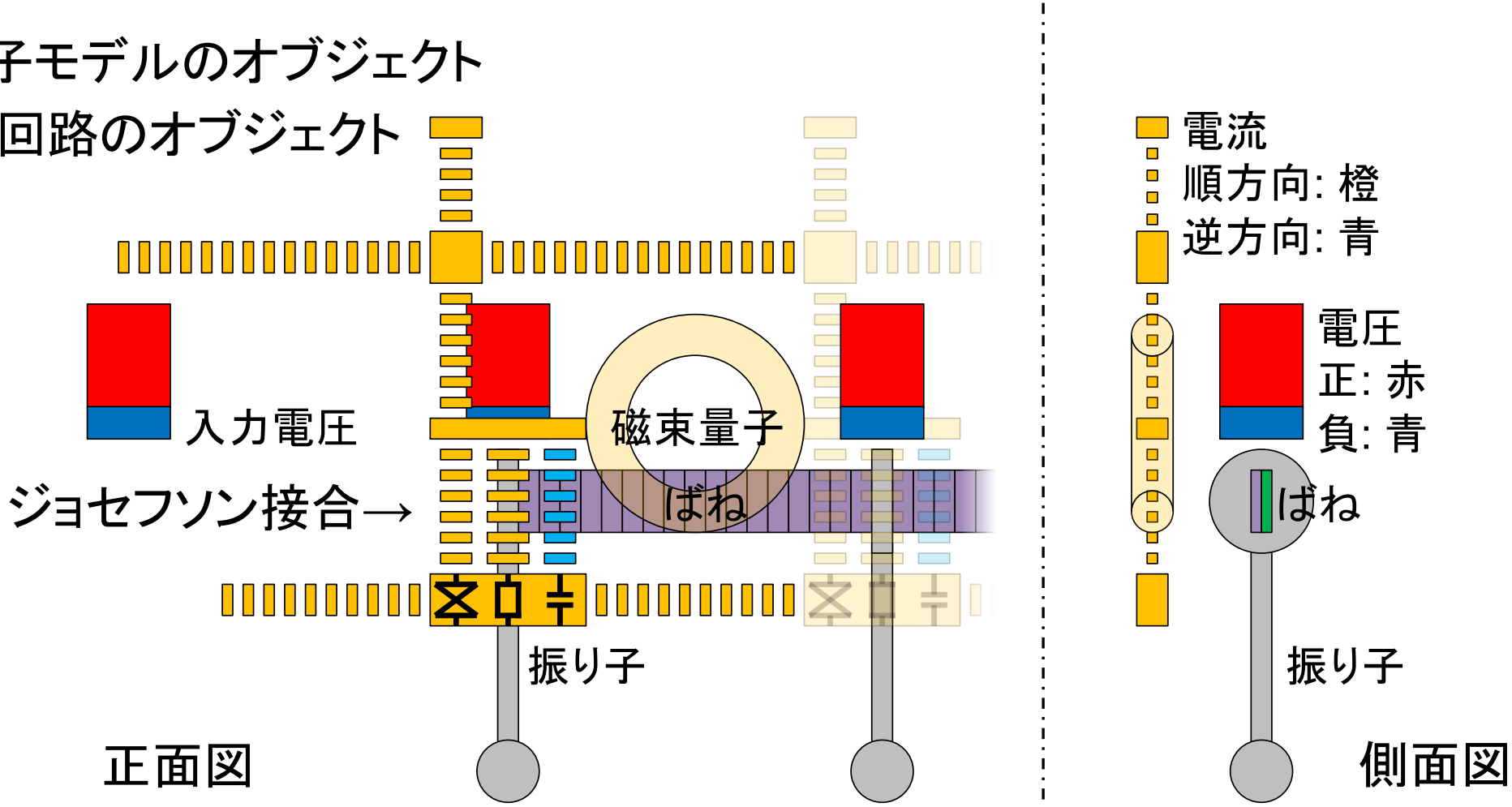
電流の表現（エイリアシング対策）

- どちら向きに動いているか？
  - 色やサイズで区別



可視化要素の配置

- xeoglベースでオブジェクトを配置する（全体像）



メニュー生成（dat.gui）: カメラワークの例

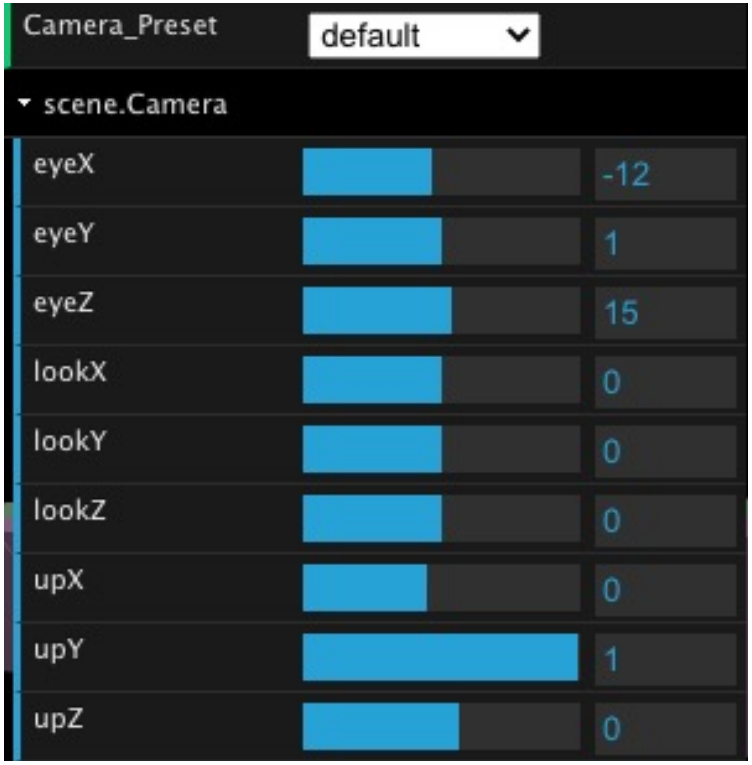
```
var CameraMenu = function () {
        this["Camera_Preset"] = "default";
        this.eyeX = camera.eye[0];
        this.eyeY = camera.eye[1];
        this.eyeZ = camera.eye[2];
        …
        var self = this;
        var lastPreset;
        var update = function () {
                var Preset = self["Camera_Preset"];
                if (lastPreset !== Preset) {
                        lastPreset = Preset;
                        var arg = CameraPresets[Preset];
                        self["eyeX"] = arg.eyeX;
                        self["eyeY"] = arg.eyeY;
                        self["eyeZ"] = arg.eyeZ;
                        camera.eye = [arg.eyeX, arg.eyeY, arg.eyeZ];
                        …
                }
                self["eyeX"] = camera.eye[0];
                self["eyeY"] = camera.eye[1];
                self["eyeZ"] = camera.eye[2];
                …
                requestAnimationFrame(update);
        }
        update();
};
```

動的な値の表示・設定

```
var CameraPresets = [];
CameraPresets["default"] = {
        eyeX: -40,
        eyeY: 10,
        eyeZ: 30,
        lookX: 0,
        lookY: 0,
        lookZ: 0,
        upX: 0,
        upY: 1,
        upZ: 0
};
```

プリセット

メニューの
レイアウトは
自動生成



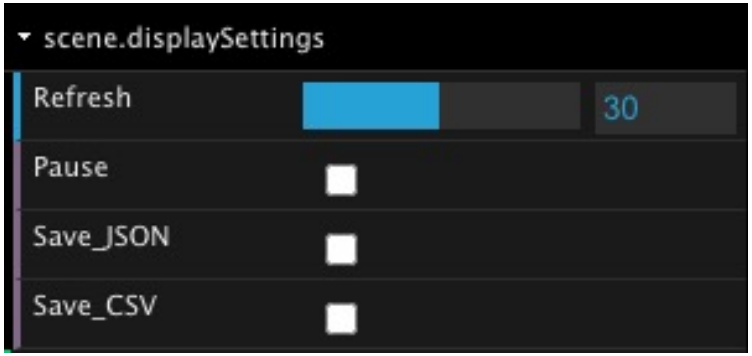

```
var camera_menu = new CameraMenu;
gui.add(camera_menu, 'Camera_Preset', Object.keys(CameraPresets));
var cameraFolder = gui.addFolder('scene.Camera' );

cameraFolder.add(camera_menu, 'eyeX', -200,200).listen();
cameraFolder.add(camera_menu, 'eyeY', -200,200).listen();
cameraFolder.add(camera_menu, 'eyeZ', -200,200).listen();
….
```

メニュー作成

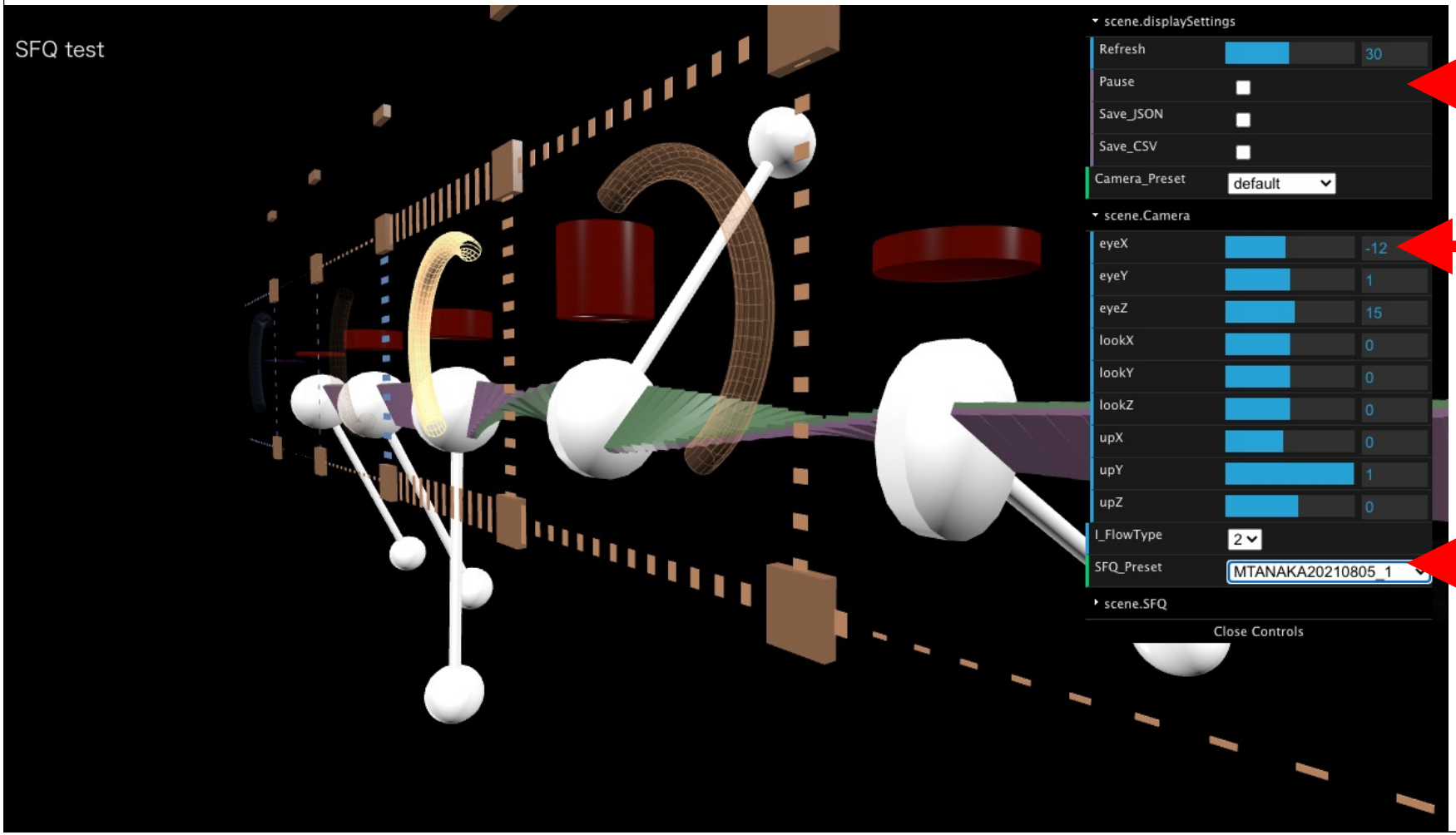
データのI/O

- 2種類検討
- CSV形式
  - データ交換用
  - 解析ソフト等との親和性
- JSON形式
  - データ構造を維持したまま書き出せる
  - JavaScriptからデータ構造へのアクセスが容易
  - Web (JavaScript)ベースで考えるなら良い選択肢



実行例

- カメラワーク: 近景、簡易表示、取り込みデータ(CSV)の可視化



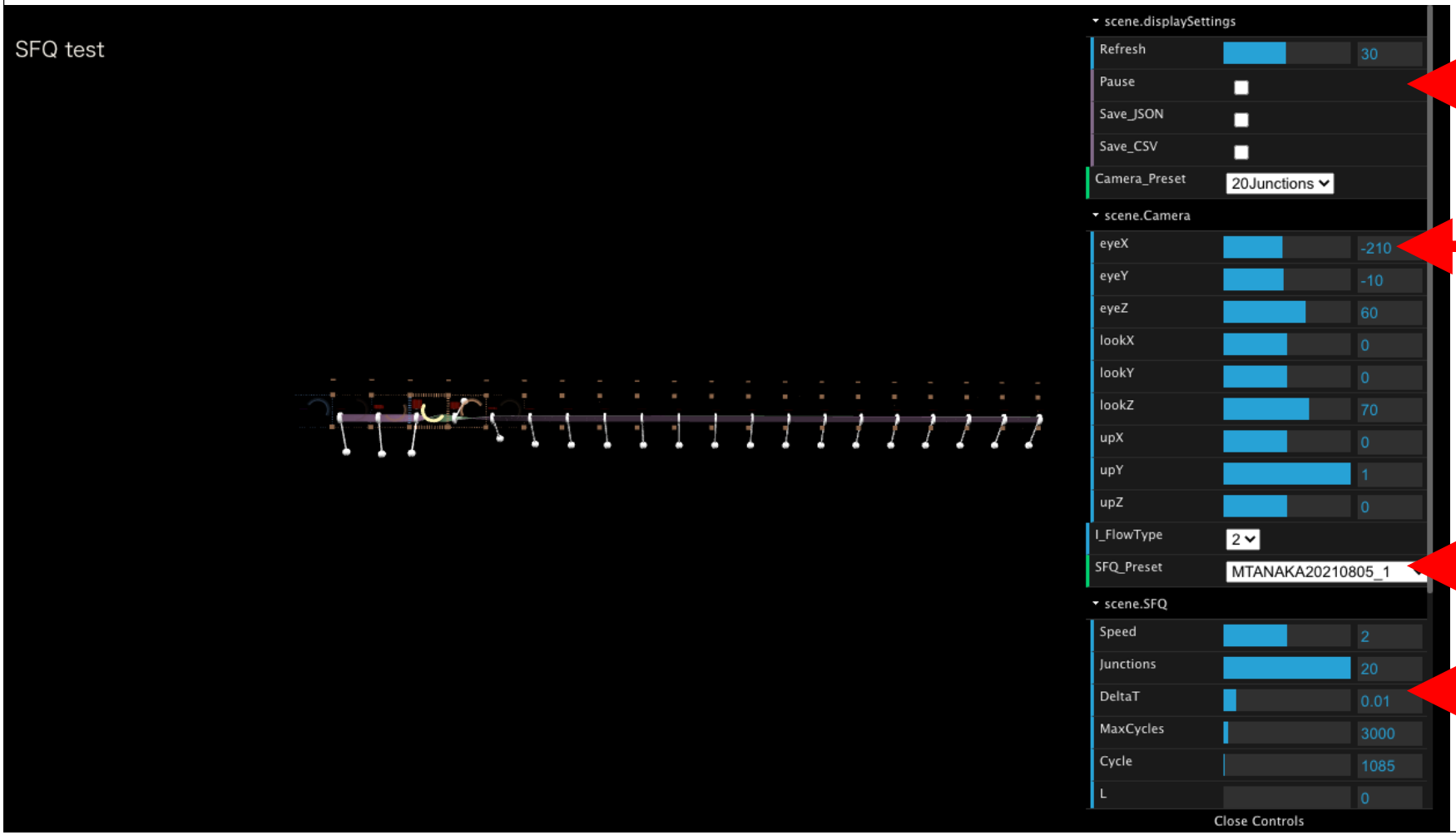
表示、I/O

カメラの
パラメータ

外部データ

実行例

- カメラワーク: 遠景、簡易表示、取り込みデータ(CSV)の可視化



表示、I/O

カメラの
パラメータ

外部データ

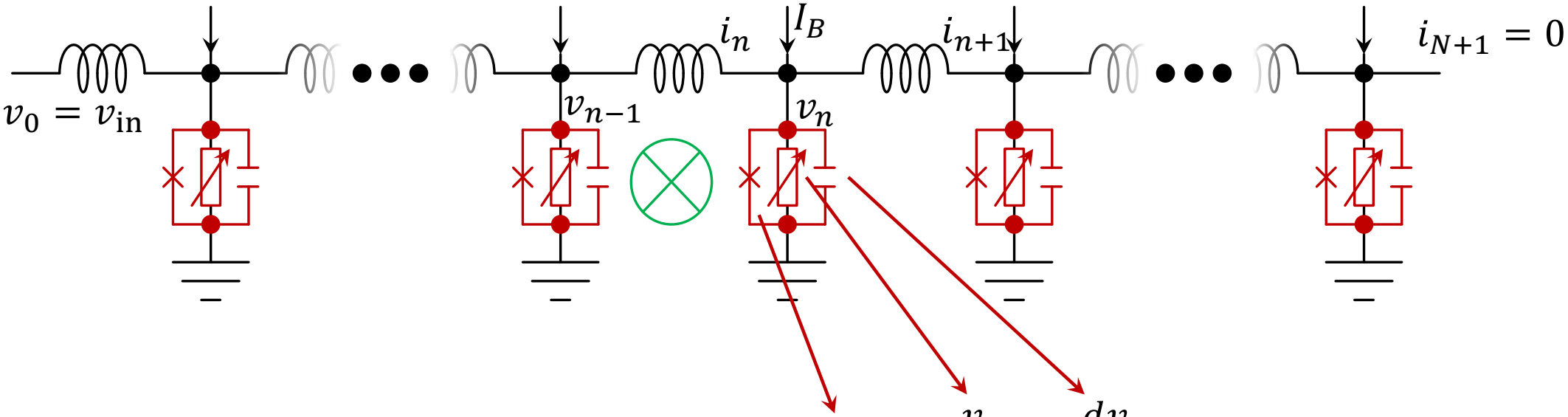
演算設定

内部計算（簡易の数値積分機能）

- 可視化に用いた回路に特化した数値積分
  - Webブラウザ内で動的に生成
- 手順
  - N段の伝搬回路（$n = 1, \ldots, N$）にて、回路方程式を考える。
  - 回路方程式を無次元化する
  - 無次元化した回路方程式を解く
    - オイラー法
    - ルンゲ・クッタ法（RK4）
- 今回用いた回路構成であれば，実用的な範囲
  - JavaScriptで現実的な時間内に計算可能
    - 体感的な待ち時間: 1秒程度?（演算＋形状生成）
  - 可視化に十分な精度で計算可能

簡易の数値積分機能

- N段の伝搬回路($n = 1, \ldots, N$)の回路方程式



$$I_B + i_n - i_{n+1} = I_c \sin\theta_n + \frac{v_n}{R} + C\frac{dv_n}{dt}$$

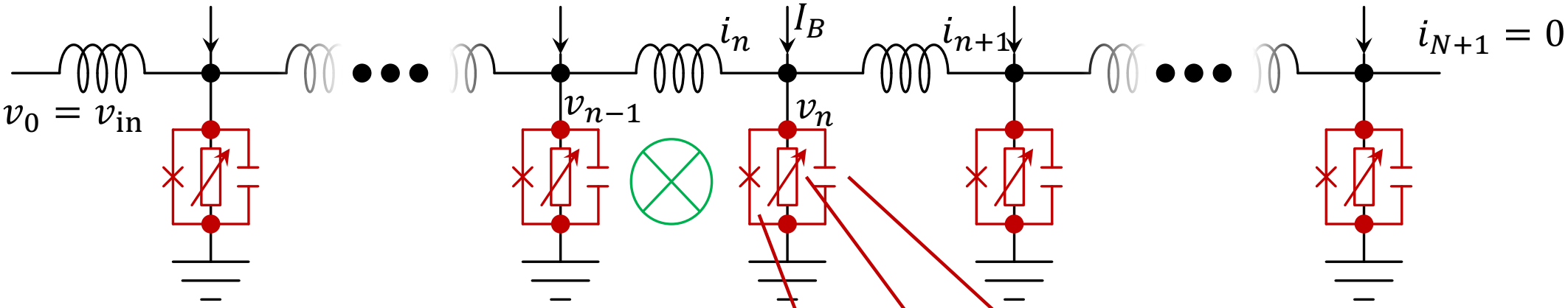
$$\Phi_0 = 2.067833\times10^{-15}\ [\mathrm{Wb}]$$

$$v_n = \frac{\Phi_0}{2\pi}\frac{d\theta_n}{dt}$$

$$v_{\mathrm{n-1}} = v_n + L_n\frac{di_n}{dt}$$

簡易の数値積分機能

- 回路方程式を無次元化する



$$T = \frac{2\pi I_c R}{\Phi_0} t$$

$$I_n = \frac{i_n}{I_c}$$

$$V_n = \frac{v_n}{I_c R}$$

$$V_0 = \frac{v_{\mathrm{in}}}{I_c R}$$

$$\frac{I_B}{I_c} + I_n - I_{n+1} = \sin\theta_n + V_n + \beta_c \frac{dV_n}{dT}$$

$$V_n = \frac{d\theta_n}{dT}$$

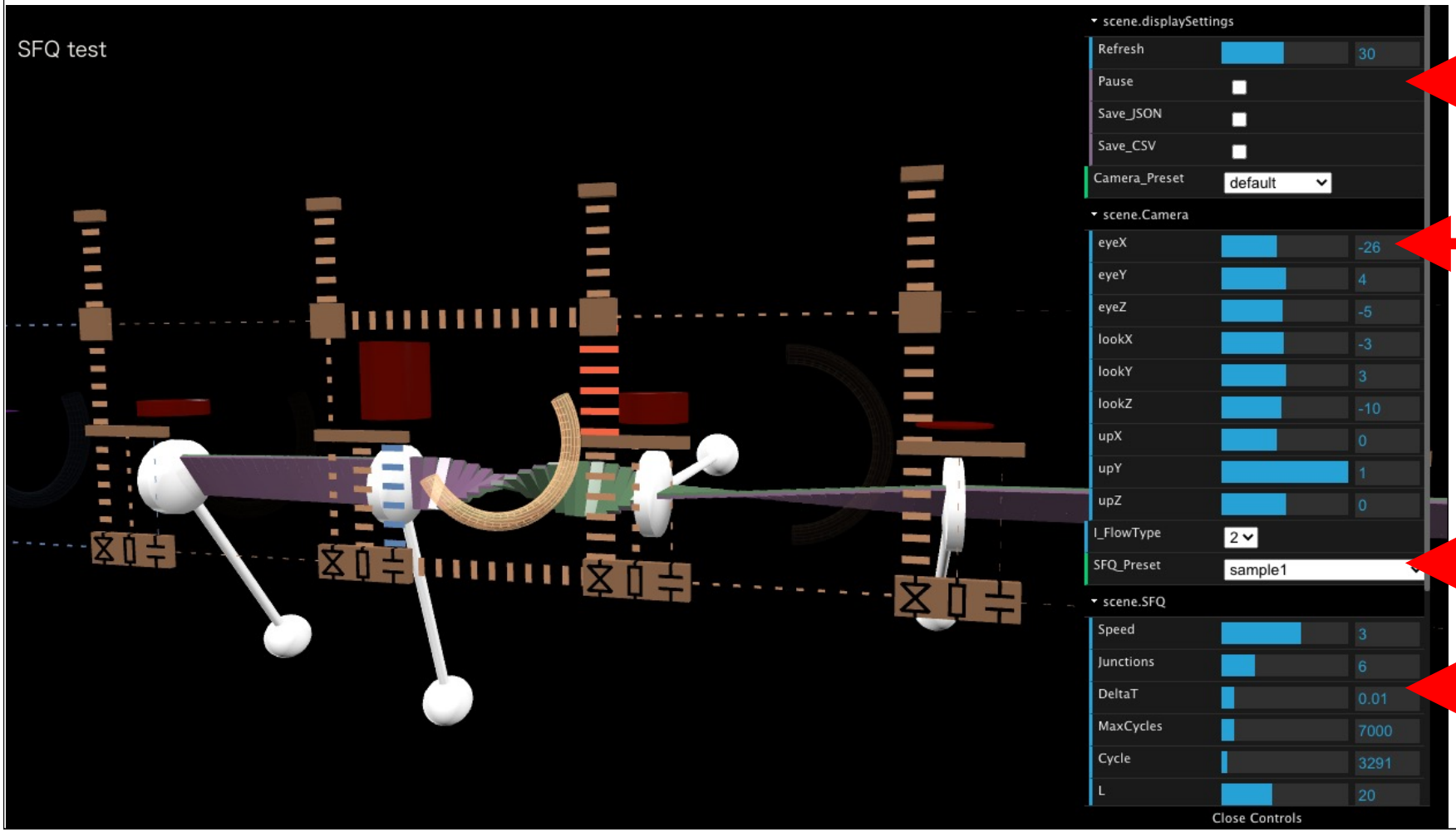
$$V_{n-1} = V_n + \frac{2\pi L_n I_c}{\Phi_0} \frac{dI_n}{dT}$$

$$I_{N+1} = 0$$

$$\beta_c = \frac{2\pi I_c R^2 C}{\Phi_0}$$

実行例

- 近景、詳細表示、内部計算（数値積分）結果の可視化



表示、I/O

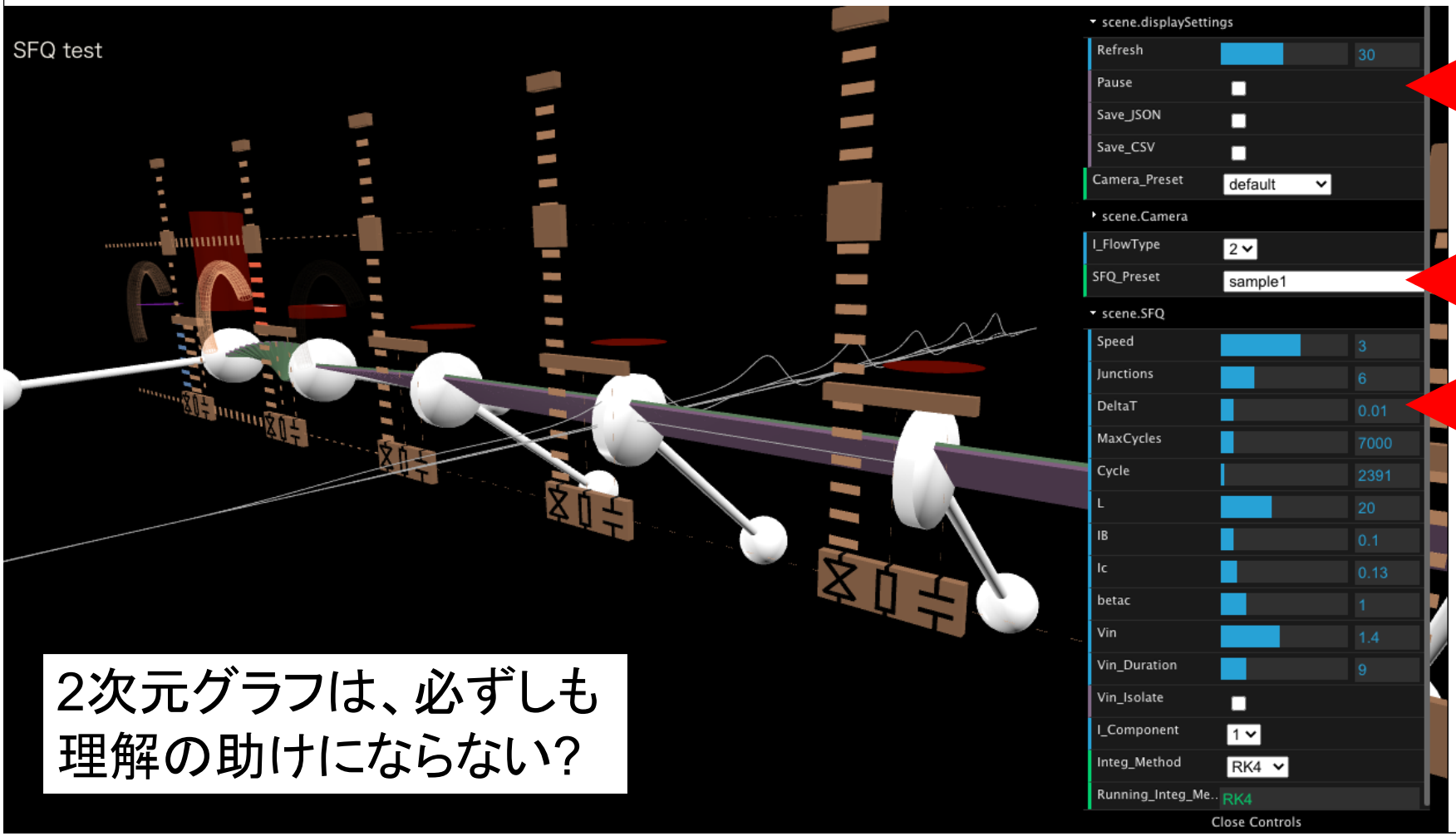
カメラの
パラメータ

外部データ

演算設定

実行例

- グラフの重ね書き（試行中）→どこまで情報を盛り込めるか



表示、I/O

外部データ

演算設定

2次元グラフは、必ずしも
理解の助けにならない？

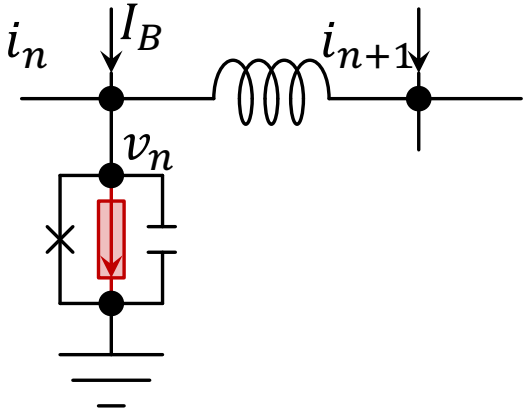
課題: ブラウザによる相違

- レンダリングの速度
  - 端末のパフォーマンスやオブジェクト数、拡大・縮小等の影響を受ける
- 線の太さ指定が反映されない?
  - xeoglの互換性の問題?
  - 線の太さを指定することができないブラウザが多い?
- 仮想マシン上のブラウザでは必ずしも正常に動作しない?

- 意図したように動作しているか，様々な環境で検証する必要がある
  - 当面は，動作確認した環境を列挙する等の対応を検討

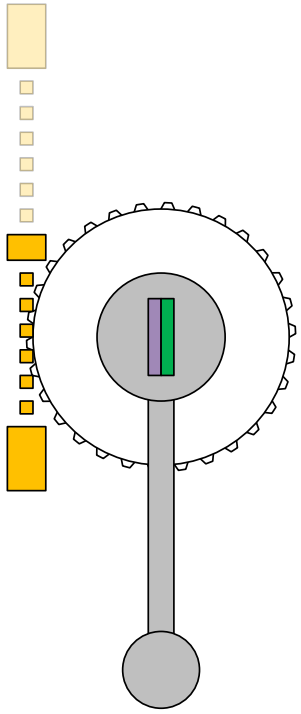
わかりやすい表現に向けて

- 例: 電流の表現
- 電流を流れの太さで表現する
  - 電流が多く流れている個所を見つけやすい
- 電流を流れの速さで表現する
  - 抵抗成分を流れる電流
    - 見かけの動きは、位相の変化（振り子の動き）と合致する.
    - モデルを工夫すると、もっとわかりやすい表現が可能かも知れない



$$\frac{I_B}{I_c} + I_n - I_{n+1} = \sin\theta_n + V_n + \beta_c \frac{dV_n}{dT}$$

$$V_n = \frac{d\theta_n}{dT}$$



今後に向けて

- 表現の改善
  - 振り子モデルは位相の変化を表現するには適している
  - 他の要素と位相変化との関連性を表現するにはさらなる工夫が必要
- 様々な種類の回路への対応
  - 現在は回路の要素ごとにソースコードを書き下している
    - 効率的に可視化するには?
    - CSV形式の入力データは、回路の種類ごとにカラム構成が変わる
  - 内蔵の数値積分機能も、回路が変われば書き換える必要がある
    - 書き換えの効率化?