Grimoire.js ハンズオン

Grimoire.js ことはじめ

Grimoire.js

諸注意

- 何か作りたいものを思いついたりしたら、聞かないで作ってもらって構いません。
- 気づいた点・疑問点などはTwitterにハッシュタグ**#grimoire_handson**で投げていただければ幸いです。

ToC

- JavascriptでGOMLを操作する

  インターフェースについて

- コンポーネントを作ってみる

  機能の最小単位

- ノードを作ってみる

  基本的にはタグの話

ハンズオン

jsからのGOMLの操作

動的にGrimoire.jsを扱うために

jsから扱う際の要点

- 操作するcanvasへのインターフェースを取得

読み込んだgomlのscript タグへのセレクタ

- 対象となるノードを取得して操作する

```
gr(function(){
  var $$ = gr("#main");
  $$("mesh").setAttribute("position", [3,0,0]);
});
```

jsから扱う際の要点

・ノードに対して可能な操作

append

remove

setAttribute

getAttribute

on

off

addComponent

ノード構造系

ノード属性系

イベント系

コンポーネント追加

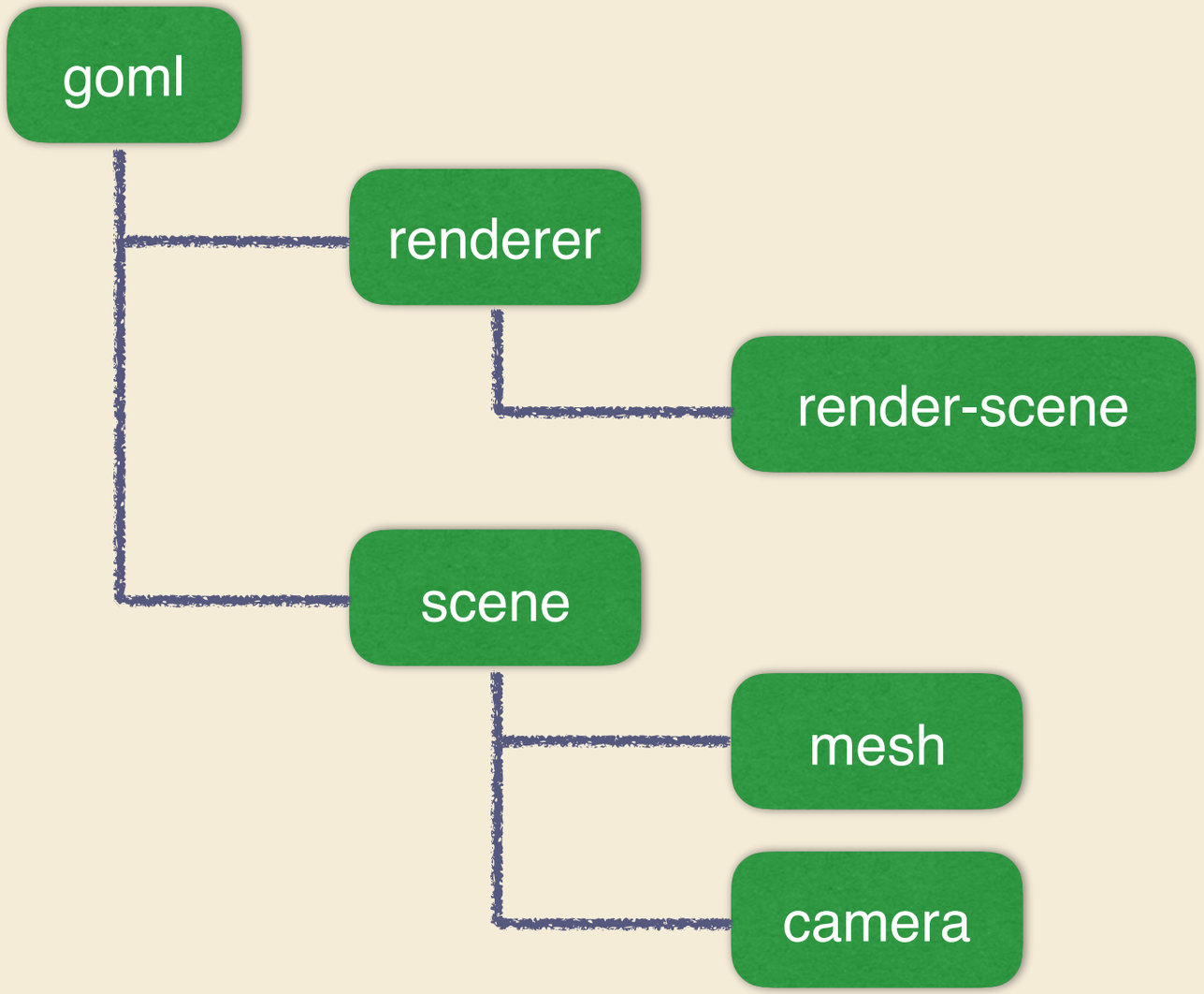
ノードとコンポーネントの本質

Grimoire.jsのデータ構造

ノードとコンポーネント

- GOMLあたり一つの木構造を持つ
  木の節々がノード

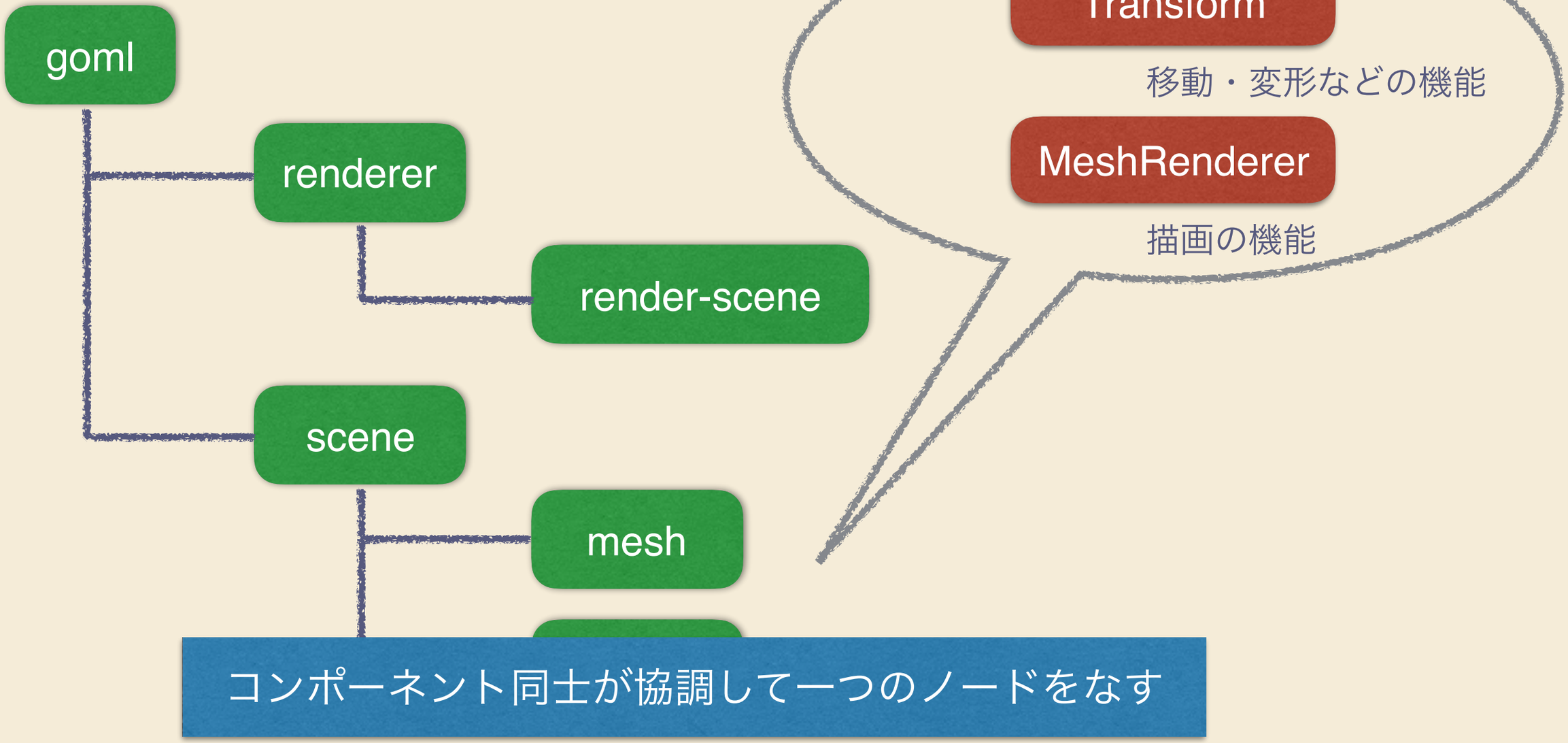
特に構造以外の機能はない




```
<goml>
    <render>
        <render-scene/>
    </render>
    <scene>
        <mesh/>
        <camera/>
    </scene>
</goml>
```

ノードとコンポーネント

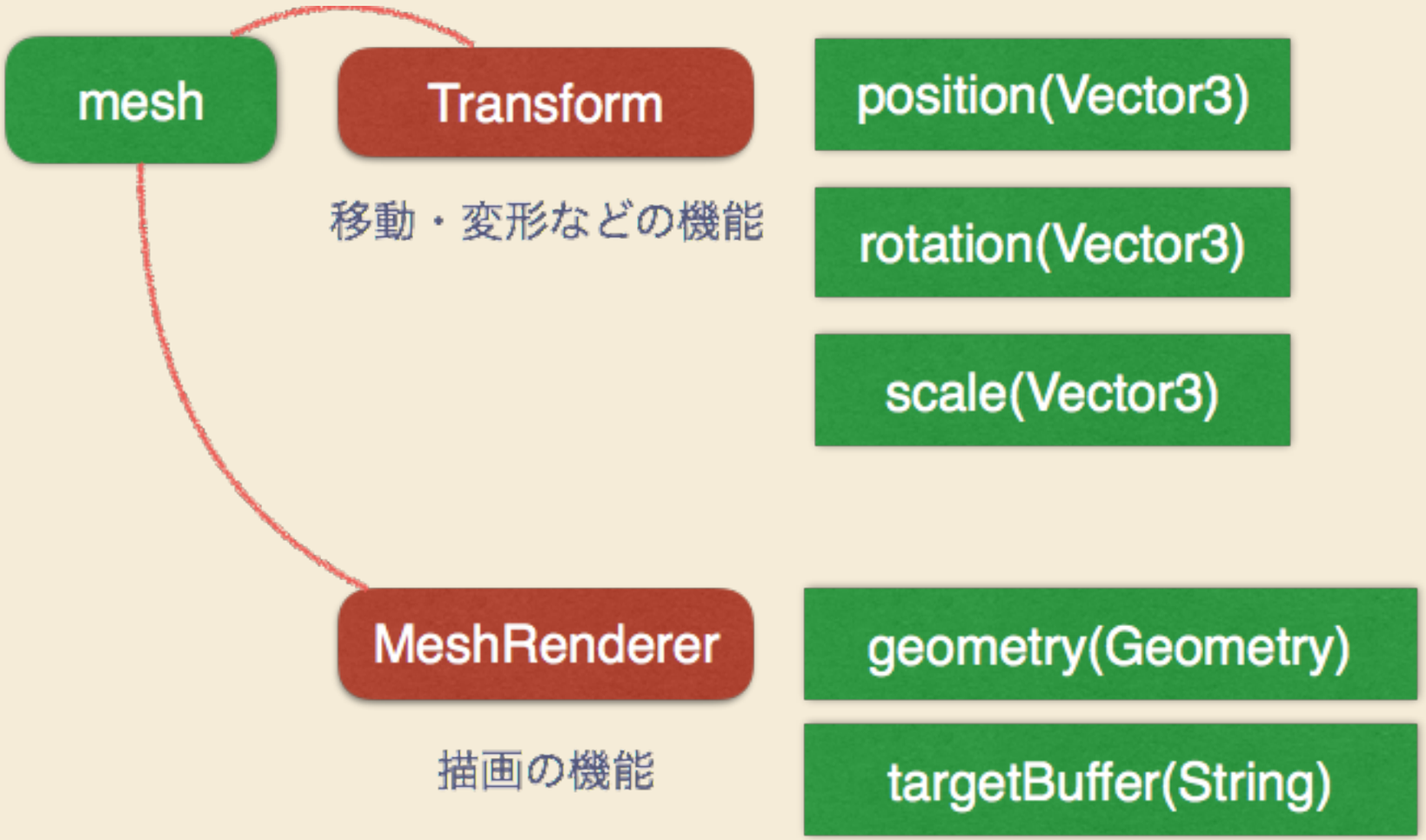
・ノードそれぞれはコンポーネントのリストを持つ



ノードとコンポーネント

- コンポーネントのそれぞれが属性を持つ

ノード名は最初に持つコンポーネントのリストに過ぎない



ノードとコンポーネントのまとめ

- ノードは**コンポーネントのリスト**を持つ木構造の一要素に過ぎない

- コンポーネントのそれぞれの属性がノードに露出

- ノード名は初期状態で生成されるコンポーネントのリストと対応

ハンズオン

コンポーネントを作ってみる

ロジックを再利用可能にする

コンポーネントの作り方の要点

- **gr.registerComponent**を用いる
- attributesにそのコンポーネントが受け取る属性を入れる。
- **$update**や**$mount**などが状況に応じて呼び出される。

コンポーネントの作り方の要点

```
gr.registerComponent("Rotate", {
    attributes: {
        speed: {
            default: 1,
            converter: "Number"
        },
        axis: {
            default: [0, 1, 0],
            converter: "Vector3"
        }
    },
    $awake: function() {
```

属性の定義

デフォルト値

コンバーター名

コンポーネントの作り方の要点

```
    },
    $mount: function() {
        this.__bindAttributes();
        this._transform = this.node.getComponent("Transform");
    },
    $update: function() {
        this._transform.localRotation = Quaternion.multiply(
            Quaternion.angleAxis(this.speed * 0.01, this.axis), this._transform.localRotation);
    }
```

RigidBodyを作成する

- Oimo.jsを組み合わせてコンポーネント作ろう
  （今回は簡単にSphereジオメトリに物理てきな挙動を付加します）

ハンズオン

ノードを作ってみる

再利用可能なノードを作成する

ノード作成の要点

・ノードの作成には**gr.registerNode**を用いる

・ノード名、初期時点でのコンポーネントの配列、初期値のリストを渡す

・ノードは**継承可能**

(初期値やコンポーネントの配列を引継ぐ)

ノード作成の要点

・ノード作成の例

```
gr.registerNode("RotationCube", ["Rotate"], {
    geometry: "cube"
}, "mesh");
```

応用例について

（セッション３に続く）