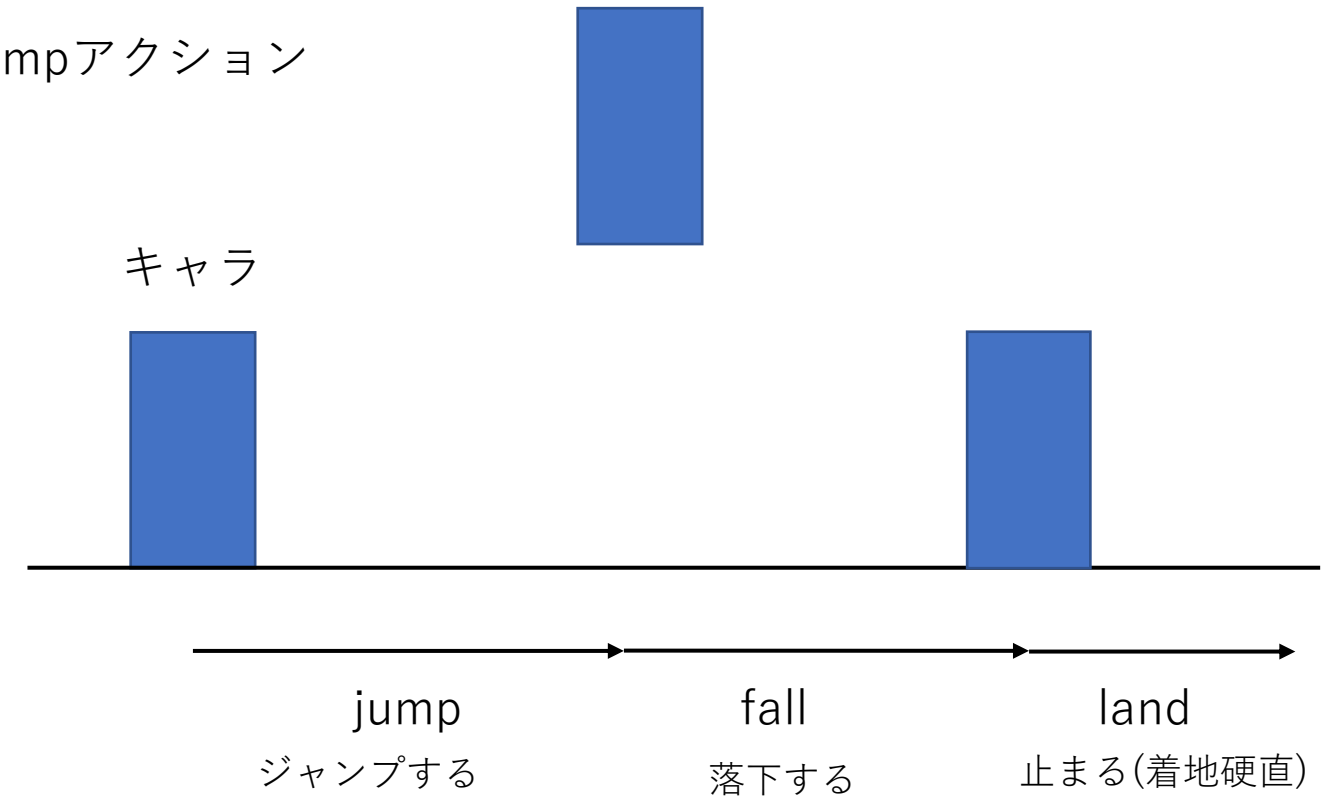
ActionManagerの使い方

Jumpアクションを作ってみる

例としてこのようなアクションを作ってみます。

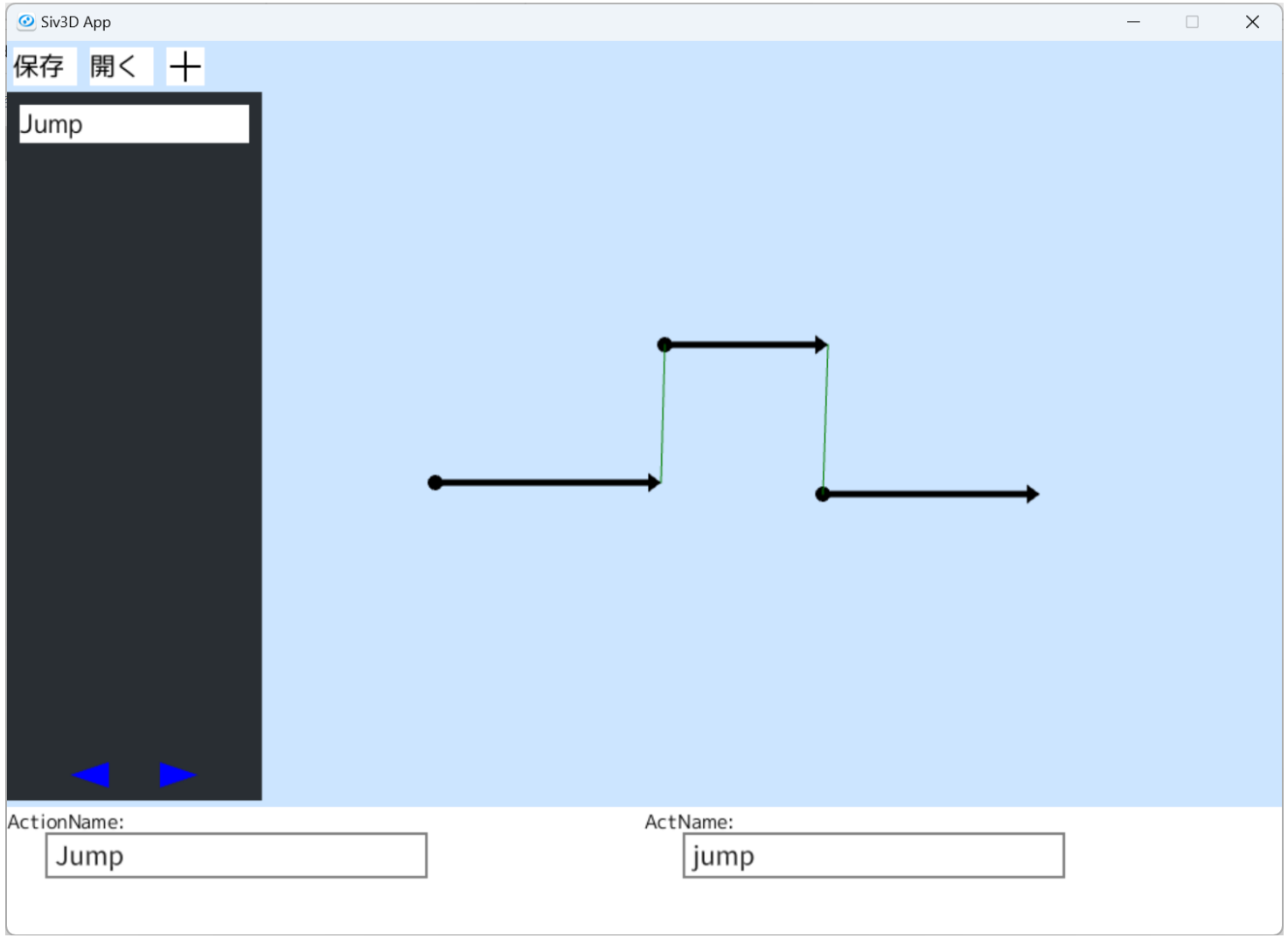


jumpはy軸方向の速度が0になるとfallに移り、地面に着地するとlandに移ります。

このアクションを作成する手順は以下の通りです。

1. ActionGUI(アクションの流れを記述したJSONファイルを作成するツール)でJSONファイル作成。
2. jump,fall,landを実行するクラスがなければ作成する。
3. ActionManagerにJSONを読み込ませて、jump,fall,landのクラスをセットする。

ActionGUI



ActionGUIはアクションの流れを記述したJSONファイル(ActionJSON)を作成するためのツールです。

別資料「ActionGUIの使い方」に使い方がのってます。

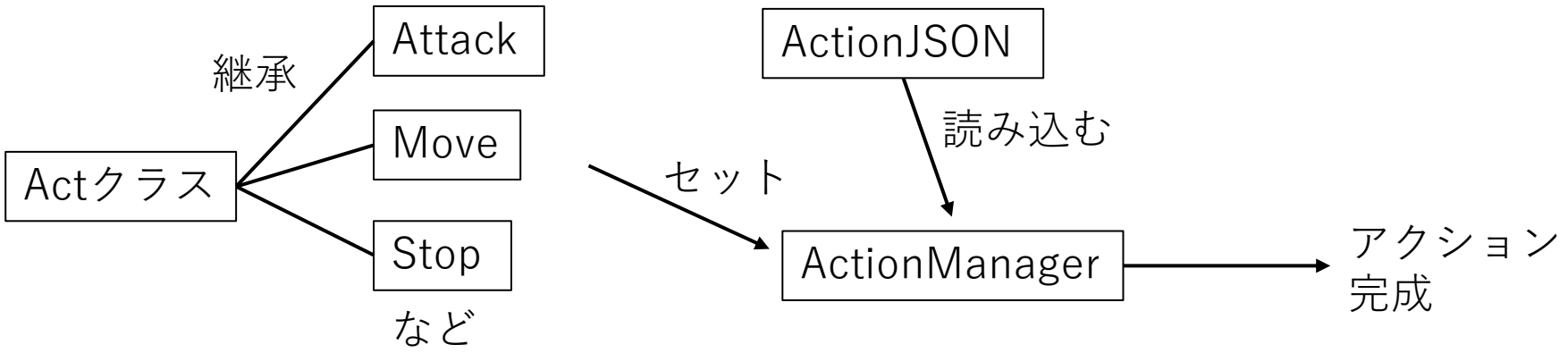
左のような矢印(Act)が配置できたら、保存してJSONファイルを作成します。

そのJSONファイルをメインプログラムのほうで読み込ませます。

ActとActionManager

- 詳しくは別資料「ActionManagerやActなど」で説明してます。

- ざっくり説明すると、Actはmoveやattackといったアクションの構成要素クラスで、ActionManagerはそのActクラスをJSON通りに実行するクラスです。



ActionManager

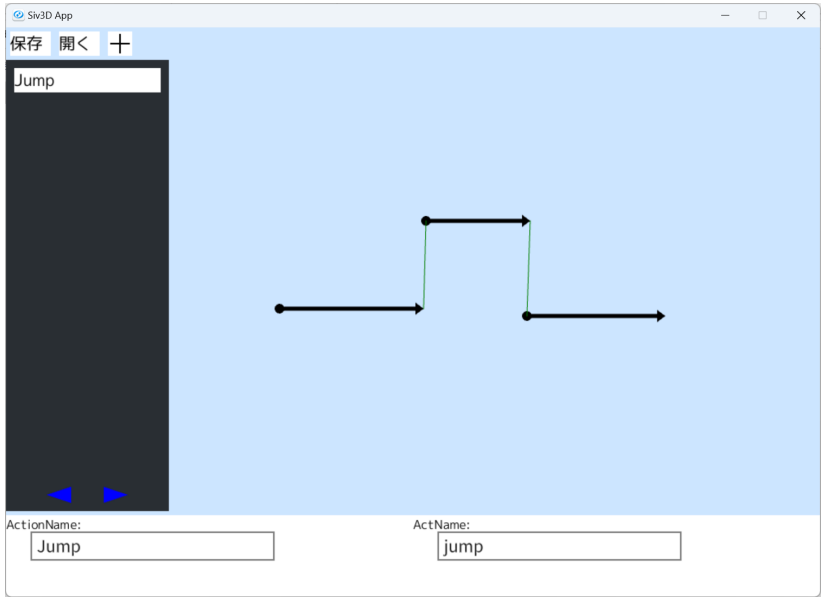
- CharacterクラスはactionsというActionManagerのオブジェクトを持ってます。
- Characterを継承したクラスがactionsを使うために、まずそのクラスのコンストラクタでactionsのセッティングを行います。
- actionsのセッティング↓

```
コンストラクタ
{
	//ActionGUIで作成したJSONファイルを読み込む
	actions->ReadJSON(U"JSONファイルのパス");

	//アクション名とアクト名を指定してActをセット
	actions->SetAct(U"アクション名", U"アクト名", Act);

	//すべてのActをセットし終えたら
	actions->SetAll();
}
```

Jumpアクションのセッティング

```
コンストラクタ
{
	//Jump(Act)を生成。Jumpの仕様でキャラクターのポインタを要求します。
	Jump* jump=new Jump(this);
	//ジャンプ中の重力(px/s^2)を1000、ジャンプの高さ(px)を200にする
	jump->SetJump(1000, 200);
	//ジャンプ後の落下アクト。これにもthisを渡す。
	Fall* fall= new Fall(this);
	//ジャンプ後の落下はジャンプ中よりも重力を大きくする。
	fall->SetG(1500);

	actions->ReadJSON(U"JSONファイルのパス");
	//"Jump"アクション"jump"アクトにjumpをセット
	actions->SetAct(U"Jump", U"jump", jump);
	//"Jump"アクション"fall"アクトにfallをセット
	actions-> SetAct(U"Jump", U"fall", fall);
	//"Jump"アクション"land"アクトにStop(Act)を生成してセット
	actions->SetAct(U"Jump", U"land", new Stop(this));
	//すべてのActをセットし終えたら
	actions->SetAll();
}
```

- ここでActの開始条件、続行条件について考えます。



・jump
開始条件: アクション開始から0秒後に開始
続行条件: 上昇が終わるまで続行
・fall
開始条件: jumpが終わってから0秒後に開始
続行条件: 地面に触れるまで続行
・land
開始条件: fallが終わってから0秒後に開始
続行条件: 1秒間続行

jumpの続行条件、fallの続行条件はActionJSONでは指定できない条件です(詳しくは別資料「ActionManagerやActなど」のActの開始条件続行条件のページにのってます)。

そのため本来なら二つの条件をキャラクターのupdate関数内で組む必要があります↓

```
if (vel.y<0)actions.EndAct(U"Jump", U"jump"); //y速度が負になったら"Jump"アクションの"jump"を終了させる。
if(box.touch(down_line))actions.EndAct(U"Jump", U"fall"); //地面についたら"Jump"アクションの"fall"を終了させる。
```

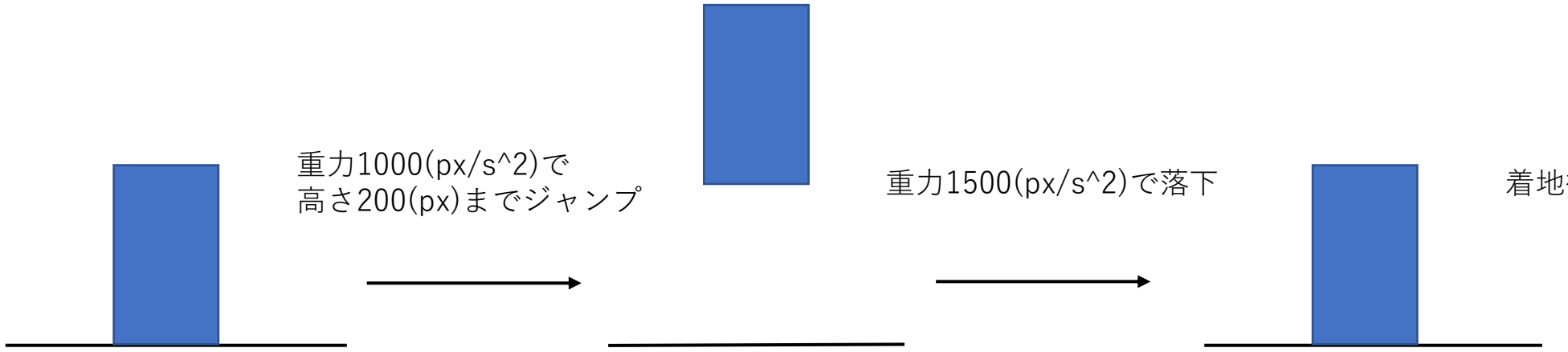
しかし、Jumpクラスにはy速度が0になると終了する仕様があり、Fallクラスには地面に触れたら終了する仕様があるので、このようなプログラムを組む必要はありません。

そのほかの条件に関してはActionGUIで指定したので、もうJumpアクション完成です。

アクションを実行する

セッティングしたアクションを実行するにはStartを呼び出します。
例↓
```
if(KeyZ.down())actions->Start(U"Jump");
```

アクションが実行出来たら、各アクトの時間やパラメータなどを調整しましょう。
各アクトの開始時間や続行時間などはJSONファイルでいじることができます。
その方法については別資料「ActionManagerやActなど」のActionJSONの編集の仕方のページを見てください。



こんな風に動くと思います(確認はしてないです)。