

如何扩展内容

如果要添加新的卡牌

比如需要添加大小王为万能牌，所有牌都可以和其进行连接。

需要改动的地方为

1. CardSuitType和CardFaceType (models/CardCommonDefine.h)

```
enum CardSuitType {
    CST_NONE = -1,
    CST_CLUBS,
    CST_DIAMONDS,
    CST_HEARTS,
    CST_SPADES,
    CST_BIGJOKER,    // 大王
    CST_SMALLJOKER,  // 小王
    CST_NUM_CARD_SUIT_TYPES
};

enum CardFaceType {
    CFT_NONE = -1,
    CFT_ACE,
    CFT_TWO,
    CFT_THREE,
    CFT_FOUR,
    CFT_FIVE,
    CFT_SIX,
    CFT_SEVEN,
    CFT_EIGHT,
    CFT_NINE,
    CFT_TEN,
    CFT_JACK,
    CFT_QUEEN,
    CFT_KING,
    CFT_JOKER,    // 王
    CFT_NUM_CARD_FACE_TYPES
};
```

2. 改变资源对应方案 (utils/Utils.h)

原本如果是花色红心那么数字对应的颜色就是红色，然后以此来寻找数字图片。

假如是方块或者红桃那么需要找的图片就是: big_red_{num}.png 和 small_red_{num}.png

所以添加花色后，资源对应方案需要更改。

```
// 花色对应颜色
static std::string getColorBySuit(CardSuitType suit) {
//  return (suit == CST_DIAMONDS || suit == CST_HEARTS) ? "red" : "black";
    if ( suit == CST_DIAMONDS || suit == CST_HEARTS || suit == CST_BIGJOKER )
        return "red";
    if ( suit == CST_CLUBS || suit == CST_SPADES || suit == CST_SMALLJOKER )
        return "black";
}
```

3. 拓展规则逻辑 (services/CardRuleService.h)

相连判断条件新增是否为大小王

```
static bool canLink(CardFaceType face1, CardFaceType face2)
{
//  return gameUtils::Utils::getFaceDistance(face1, face2) == 1;
    if (face1 == CFT_JOKER || face2 == CFT_JOKER) return true;
    return utils::getFaceDistance(face1, face2) == 1;
}
```

如何新增一个回退功能

比如新增一个按钮，可以选择消除PlayField区域的牌

假设消除逻辑的简单是

```
void PlayFieldController::removeCardBySkill(CardView* cardView) {
    // 1. 根据cardView 拿到对应的 cardModel 数据
    // 2. 将 cardModel 从PlayFieldModel区域移除
    // 3. cardView移除

    CardModel* cardModel = cardView->getCardDataPtr();
    services::FieldModelService::removeCardModelFromFieldModel(
        cardModel, _playFieldModel
    );
    cardView->retain();
    _playFieldView->removeCardView(cardView);
}
```

所以更改的信息为:

1. cardModel
2. cardView (position, zOrder)

所以如果要撤销的话需要对这两个信息进行恢复。

1. 需要拓展UndoStep为支持多种UndoData (models/UndoModel.h)

```

enum class UndoType {
    PlayToStack,
    BottomToTop,
    SkillRemoveCard // 对应的UndoType
};

/* 原本
struct UndoStep {
    UndoType type;
    int cardId;

    cocos2d::Vec2 oldPos;
    int zorder;
};
*/

// 仅移动到StackFiled顶牌对应的Undo信息
struct UndoPlayMove {
    int cardId;
    cocos2d::Vec2 oldPos;
    int zorder;
};

// 消除卡牌技能对应的Undo信息
struct UndoSkillRemove {
    int cardId;
    cocos2d::Vec2 oldPos;
    int zorder;
    models::CardModel* removedCard;
    views::CardView* removeCardView;
};

using UndoData = std::variant<UndoPlayMove, UndoSkillRemove>;

// 将UndoStep更新为支持多种UndoData
struct UndoStep {
    UndoType type;
    UndoData data;
};

```

2. 技能触发时，数据状态改变之前需要对UndoData以及UndoStep进行保存

```

void PlayFieldController::removeCardBySkill(views::CardView* cardView) {
    auto* cardModel = cardView->getCardDataPtr();

    // 保存当前数据到UndoData中
    UndoSkillRemove undoData;
    undoData.cardId = cardModel->id;
    undoData.oldPos = cardView->getPosition();
    undoData.zorder = cardView->getLocalZOrder();
    undoData.removedCard = cardModel;
    undoData.removeCardView = cardView;
}

```

```

// 保存当前UndoData到UndoStep中
UndoStep step;
step.type = UndoType::SkillRemoveCard;
step.data = undoData;

// 将UndoStep压住栈顶
_undoManager->pushStep(step);

// 执行技能消除效果
services::FieldModelService::removeCardModelFromFieldModel(
    cardModel, _playFieldModel
);
cardView->retain();
_playFieldView->removeCardView(cardView);
}

```

3. 对根据UndoType对修改的数据进行恢复

需要：

1. 向PlayFieldModel中添加移除的cardModel
2. 将原本的cardView放回到PlayFieldView中

```

void GameController::onUndoPressed() {
    ...
    // 判断栈顶元素是否是 SkillRemoveCard
    if (step.type == models::UndoType::SkillRemoveCard) {
        _playFieldController->removeCardBySkill(step);
    }
    ...
}

void PlayFieldController::undoRemoveCardBySkill(
    const UndoSkillRemove& data
) {
    // 恢复 Model
    services::FieldModelService::addCardModelToFieldModel(
        data.removedCard, _playFieldModel
    );

    // 恢复原本被隐藏的 View
    auto* cardView = data.removedView;
    _playFieldView->addChild(cardView, data.zorder);
    cardView->setPosition(data.oldPos);

    // 抵消之前的 retain
    cardView->release();
}

```

