Novas Funcionalidades adicionadas a Ruined Lost City Rumours

- **Give Boat**

Na ocorrência deste evento o utilizador recebe um novo Barco de qualquer tipo, ao jogador sendo este entregue ao porto Europeu do mesmo.

Esta Feature traz uma dinâmica maior ao jogo dando a possibilidade de ganhar qualquer barco desde o mais barato ao mais raro, ajudando bastante o jogador.

Implementação

```
case GIVE_BOAT:
    ServerEurope playerEurope = (ServerEurope)owner.getEurope();
    if (playerEurope == null) {
        // FoY should now be disabled for non-colonial
        // players, but leave this in for now as it is harmless.
        cs.addMessage(owner,
                new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                        rumour.getAlternateDescriptionKey( variant: "noEurope"),
                        owner,  display: this));
    } else {
        Unit u = playerEurope.generateFreeBoat(random);
        cs.add(See.only(owner), playerEurope);
        cs.addMessage(owner,
                new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                        key, owner, u)
                        .addName( key: "%boat%", u.getType()));
    }
    break;
```

```
public Unit generateFreeBoat(Random random) {
    final Game game = getGame();
    final Player owner = getOwner();
    List<RandomChoice<UnitType>> boats = generatePossibleBoatsList();
    UnitType ut = RandomChoice.getWeightedRandom(logger,  logMe: "Choose FoY",
            boats, random);
    return new ServerUnit(game,  location: this, owner, ut);
}
```

```
private List<RandomChoice<UnitType>> generatePossibleBoatsList() {
    return transform(getSpecification().getUnitTypeList(),
            ut ->   ut.isNaval(),
            ut -> new RandomChoice<>(ut,  probability: 100));
}
```

```
model.lostCityRumour.giveBoat.description= A %boat% has been delivered to your port in Europe!
```

É gerada uma lista com todos os barcos possíveis de serem atribuídos ao utilizador (sendo que neste caso o barco gerado pode ser qualquer um existente no jogo com a habilidade NAVAL_UNIT, todos com a mesma chance de calhar). É escolhido o barco em concreto e criada a nova instância de ServerUnit, e esta atribuída ao utilizador no seu porto Europeu.

É ainda adicionada uma mensagem como mostra a primeira imagem, usando o modelo pré-existente de mensagem de LOST_CITY_RUMOUR, enviando o objeto (Unit u) barco para obter a imagem do mesmo, o dono (owner) a quem se deve atribuir o barco e a DescriptionKey (key), para obter a Descrição criada em data/strings/FreeColMessages.properties. É ainda substituída a palavra %boat% na mensagem pelo tipo de barco recebido pelo utilizador

- **Give Wagon**

Na ocorrência deste evento o utilizador recebe um novo Vagão

Esta Feature ajuda a mobilidade entre cidades do utilizador, apesar de ser uma das piores que podem sair no rumor comparativamente às outras.

Implementação

```
case GIVE_WAGON:
    List<UnitType> carry
            = spec.getUnitTypesWithAbility(Ability.ONLY_GOODS);
    unitType = getRandomMember(logger,  logMe: "Choose Wagon ",
            carry, random);
    newUnit = new ServerUnit(game, tile, owner,
            unitType);
    cs.addMessage(owner,
            new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                    key, owner, newUnit).addName( key: "%wagon%", newUnit.getType()));
    break;
```

```
/** The ability of certain units (e.g. wagon trains) to carry goods. */
public static final String ONLY_GOODS
        = "model.ability.onlyGoods";
```

```
<unit-type id="model.unit.wagonTrain" extends="wagon"
          offence="0" defence="1" movement="6"
          score-value="1" spaceTaken="12" space="2">
  <required-goods id="model.goods.hammers"
                  value="40"/>
  <limit id="model.limit.wagonTrains" operator="lt">
    <left-hand-side operand-type="units" scope-level="player" />
    <right-hand-side operand-type="settlements" scope-level="player" />
  </limit>
  <ability id="model.ability.carryGoods"
          value="true"/>
  <ability id="model.ability.onlyGoods"
          value="true"/>
  <ability id="model.ability.canBeCaptured"
          value="true"/>
</unit-type>
```

```
model.lostCityRumour.giveWagon.description= You have found a new %wagon%
```

É utilizada uma função pre-existente que retorna uma lista de unidades com uma certa abilidade atribuída no ficheiro specification.xml em data/rules/classic, neste caso criamos as habilidades na Classe Ability, e atribuímos as habilidades no xml de forma a ser retornada apenas o vagão por nós requisitado pela função, neste caso Ability.ONLYGOODS, já que este apenas consegue carregar bens e não pessoas ao contrário dos barcos. É atribuída a nova unidade ao utilizador com a função Server Unit fornecendo a tile associada ao rumor (local de spawn) e o utilizador a atribuir. É novamente adicionada uma mensagem em que é substituído %wagon% da Descrição criada em data/strings/FreeColMessages.properties pelo tipo da unidade.

- **Give Armed Boat**

Ao entrar numa cidade perdida arruinada, ao calhar este evento aleatório, o utilizador que explorou a cidade recebe recursos (cavalos e mosquetes). Estes são entregues ao porto do utilizador, sendo estes entregues em uma caravela, para que mais tarde possa-se transferir os recursos para colónias do utilizador

Implementação

```
case GIVE_ARMED_BOAT:
    ServerEurope pEurope = (ServerEurope)owner.getEurope();
    if (pEurope == null) {
        cs.addMessage(owner,
                new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                        rumour.getAlternateDescriptionKey( variant: "noEurope"),
                        owner,  display: this));
    } else {
        Unit u = pEurope.generateFreeArmedBoat(random);
        cs.add(See.only(owner), pEurope);
        cs.addMessage(owner,
                new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                        key, owner, u)
                        .addName( key: "%boat%", u.getType())
                        .addName( key: "%equipment1%", u.getGoodsContainer().getGoodsList().get(0))
                        .addName( key: "%equipment2%", u.getGoodsContainer().getGoodsList().get(1))
                        .addAmount( key: "%q1%", u.getGoodsContainer().getGoodsList().get(0).getAmount())
                        .addAmount( key: "%q2%", u.getGoodsContainer().getGoodsList().get(1).getAmount()));
        cs.addAttribute(See.only(owner),
                 key: "sound",  value: "sound.attack.artillery");
    }

    break;
```

```
public Unit generateFreeArmedBoat(Random random) {
    final Game game = getGame();
    final Player owner = getOwner();
    Specification spec = game.getSpecification();
    List<RandomChoice<UnitType>> boats = generatePossibleBoatsCaravel();
    UnitType ut = RandomChoice.getWeightedRandom(logger,  logMe: "Choose FoY",
            boats, random);
    Unit boat = new ServerUnit(game,  location: this, owner, ut);
    GoodsType t1 = spec.getGoodsType( id: "model.goods.horses");
    GoodsType t2 = spec.getGoodsType( id: "model.goods.muskets");
    ((ServerPlayer) owner).stealInEurope(random,boat.getGoodsContainer(),t1,  amount: 100);
    ((ServerPlayer) owner).stealInEurope(random,boat.getGoodsContainer(),t2,  amount: 100);
    return boat;
}
```

```
private List<RandomChoice<UnitType>> generatePossibleBoatsCaravel() {
    return transform(getSpecification().getUnitTypeList(),
            ut ->   ut.isNaval() && ut.hasAbility(Ability.CARAVEL),
            ut -> new RandomChoice<>(ut,  probability: 100));
}
```

```
<unit-type id="model.unit.caravel" extends="ship" price="1000"
           offence="0" defence="2" movement="12"
           line-of-sight="1" space="2" spaceTaken="24" hit-points="6"
           score-value="3">
  <required-goods id="model.goods.hammers"
                  value="128"/>
  <required-goods id="model.goods.tools"
                  value="40"/>
  <ability id="model.ability.caravel"
           value="true"/>
```

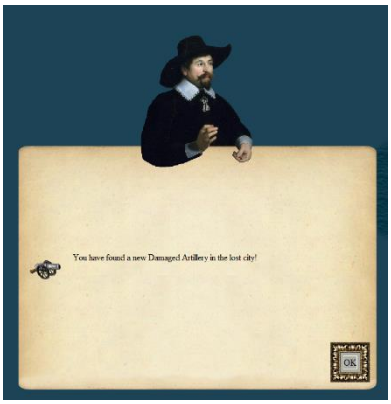
```
public static final String CARAVEL
        = "model.ability.caravel";
```

```
model.lostCityRumour.giveArmedBoat.description= A %boat% has been delivered to your port in Europe, equipped with %q1% %equipment1% and %q2% %equipment2%!
```

No case exemplificado, o método executa 2 processos diferentes. Ele gera uma caravela (através dos métodos generateFreeArmedBoat e generatePossibleBoatsCaravel [este método que assegura se que, da lista de barcos que devolve, apenas aceita aquele com a ability.caravel), equipada com resources (100 de muskets e 100 de horses) (através do método stealInEurope ao utilizador, fazendo com que esta seja devolvida ao porto do utilizador.

É de seguida criada a mensagem tal como a imagem demosntra, usando o modelo pré existente de mensagem LOST_CITY_RUMOUR. No comando addMessage, é necessário usar a Unit u para determinar a imagem do navio usado (neste caso, a Caravela), o dono e a descriptionKey pois esta é que determina o formato da mensagem (demonstrada nas imagens), onde certas partes (como os %equipment1% e %q1%) são substituídos pelo tipo de resource e quantidade, respetivamente

- **Give Artillery/ Give Damaged Artillery**



Na ocorrência destes evento o utilizador recebe uma nova Artilharia na própria tile, no caso de GIVE_DAMAGED_ARTILERY recebe uma artilharia menos potente que até vai ser utilizada como algo mau que pode calhar nos Ruined Lost city rumours por ser muito fraca, enquanto que GIVE_ARTILLERY dá uma artilharia normal, é uma feature bastante potente pois evita ter de se traze-la da capital, nem pagar por ela.

Implementação

```
case GIVE_ARTILLERY:
    List<UnitType> bombardUnitTypes
            = spec.getUnitTypesWithAbility(Ability.NORMAL_BOMBARD);
    unitType = getRandomMember(logger, logMe: "Choose artillery",
            bombardUnitTypes, random);
    newUnit = new ServerUnit(game, tile, owner,
            unitType);
    cs.addMessage(owner,
            new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                    key, owner, newUnit).addName( key: "%piece%", newUnit.getType()));
    cs.addAttribute(See.only(owner),
             key: "sound",  value: "sound.attack.artillery");
    break;

case GIVE_DAMAGED_ARTILLERY:
    List<UnitType> damagedBombardUnitTypes
            = spec.getUnitTypesWithAbility(Ability.DAMAGED_BOMBARD);
    unitType = getRandomMember(logger, logMe: "Choose artillery",
            damagedBombardUnitTypes, random);
    newUnit = new ServerUnit(game, tile, owner,
            unitType);
    cs.addMessage(owner,
            new ModelMessage(ModelMessage.MessageType.LOST_CITY_RUMOUR,
                    key, owner, newUnit).addName( key: "%piece%", newUnit.getType()));
    cs.addAttribute(See.only(owner),
             key: "sound",  value: "sound.attack.artillery");
    break;
```

```
</unit-type>
<unit-type id="model.unit.artillery" extends="wagon" price="500"
          offence="7" defence="5" movement="3"
          score-value="2" hit-points="6">
  <required-goods id="model.goods.hammers"
                  value="192"/>
  <required-goods id="model.goods.tools"
                  value="40"/>
  <ability id="model.ability.refUnit"
           value="true"/>
  <ability id="model.ability.canBeSurrendered"
           value="true"/>
  <ability id="model.ability.captureUnits"
           value="true"/>
  <ability id="model.ability.mercenaryUnit"
           value="true"/>
  <ability id="model.ability.supportUnit"
           value="true"/>
  <ability id="model.ability.bombard"
           value="true"/>
  <ability id="model.ability.normalBombard"
           value="true"/>
</unit-type>
```

```
/** The ability of a unit to bombard other units. */
public static final String BOMBARD
        = "model.ability.bombard";
/** The ability of a unit to bombard other units. */
public static final String DAMAGED_BOMBARD
        = "model.ability.damagedBombard";
/** The ability of a unit to bombard other units. */
public static final String NORMAL_BOMBARD
        = "model.ability.normalBombard";
```

```
<unit-type id="model.unit.damagedArtillery" extends="wagon"
          offence="5" defence="3" movement="3"
          score-value="1" hit-points="6">
  <ability id="model.ability.captureUnits"
           value="true"/>
  <ability id="model.ability.bombard"
           value="true"/>
  <ability id="model.ability.damagedBombard"
           value="true"/>
</unit-type>
```

```
model.lostCityRumour.giveArtillery.description= You have found a new %piece% in the lost city!
```

Em ambos os métodos é utilizado uma função pre-existente que retorna uma lista de unidades com uma certa abilidade atribuída no ficheiro specification.xml em data/rules/classic, neste caso criamos as habilidades na Classe Ability, e atribuímos as habilidades no xml de forma a ser retornada apenas a artilharia por nós requisitada pela função, neste caso Ability.NORMAL_BOMBARD para a artilharia normal e Ability.DAMAGED_BOMBARD para a artilharia danificada. É atribuída a nova unidade ao utilizador com a função Server Unit fornecendo a tile associada ao rumor (local de spawn) e o utilizador a atribuir. É novamente adicionada uma mensagem em que é substituído %piece% da Descrição criada em data/strings/FreeColMessages.properties pelo tipo da unidade em ambos os casos.