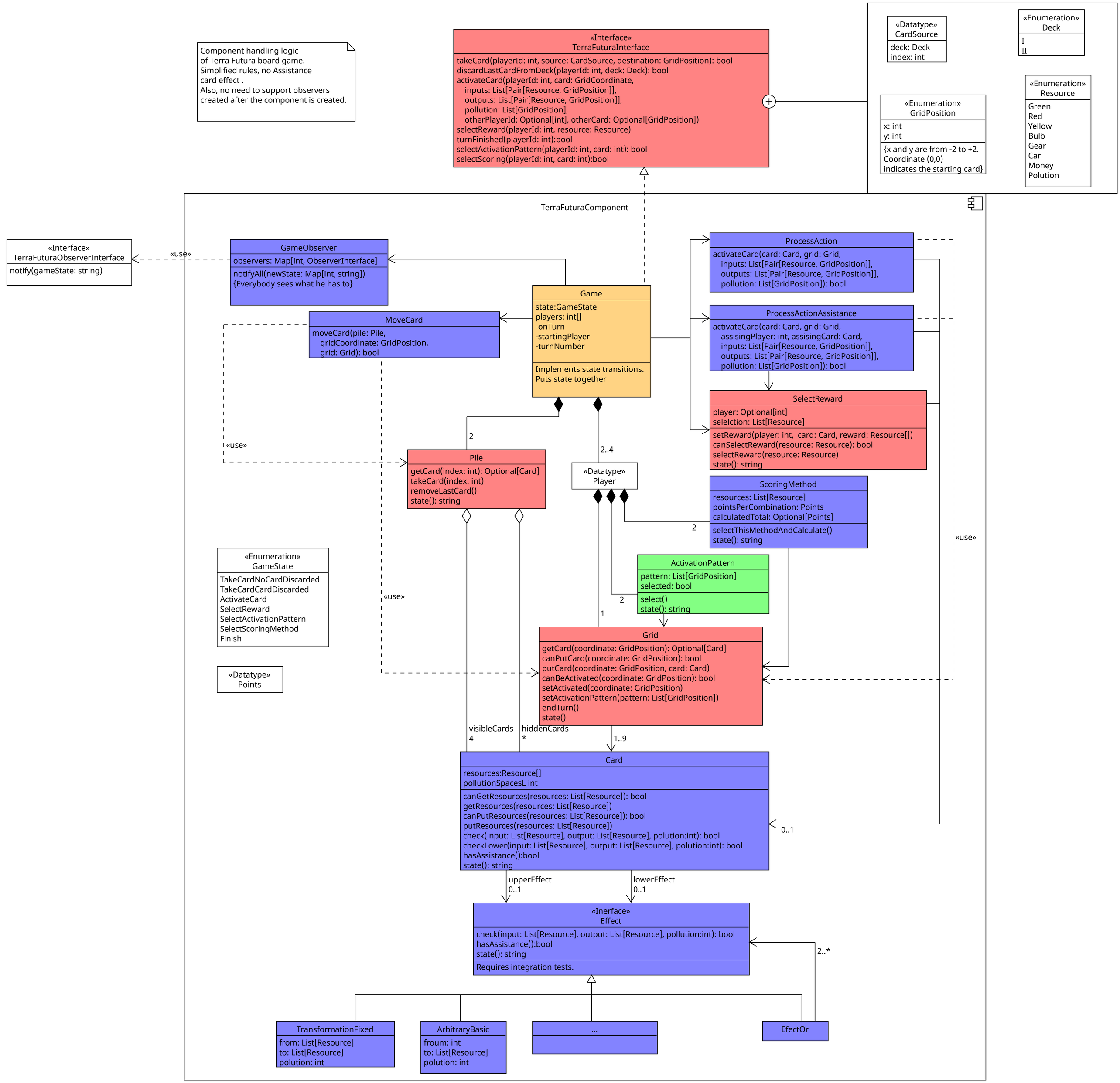
Component handling logic
of Terra Futura board game.
Simplified rules, no Assistance
card effect .
Also, no need to support observers
created after the component is created.
«Interface»
TerraFuturaInterface
takeCard(playerId: int, source: CardSource, destination: GridPosition): bool
discardLastCardFromDeck(playerId: int, deck: Deck): bool
activateCard(playerId: int, card: GridCoordinate,
inputs: List[Pair[Resource, GridPosition]],
outputs: List[Pair[Resource, GridPosition]],
pollution: List[GridPosition],
otherPlayerId: Optional[int], otherCard: Optional[GridPosition])
selectReward(playerId: int, resource: Resource)
turnFinished(playerId: int):bool
selectActivationPattern(playerId: int, card: int): bool
selectScoring(playerId: int, card: int):bool
«Datatype»
CardSource
deck: Deck
index: int
«Enumeration»
Deck
I
II
«Enumeration»
GridPosition
x: int
y: int
{x and y are from -2 to +2.
Coordinate (0,0)
indicates the starting card}
«Enumeration»
Resource
Green
Red
Yellow
Bulb
Gear
Car
Money
Polution
TerraFuturaComponent
«Interface»
TerraFuturaObserverInterface
notify(gameState: string)
«use»
GameObserver
observers: Map[int, ObserverInterface]
notifyAll(newState: Map[int, string])
{Everybody sees what he has to}
Game
state:GameState
players: int[]
-onTurn
-startingPlayer
-turnNumber
Implements state transitions.
Puts state together
MoveCard
moveCard(pile: Pile,
gridCoordinate: GridPosition,
grid: Grid): bool
ProcessAction
activateCard(card: Card, grid: Grid,
inputs: List[Pair[Resource, GridPosition]],
outputs: List[Pair[Resource, GridPosition]],
pollution: List[GridPosition]): bool
ProcessActionAssistance
activateCard(card: Card, grid: Grid,
assisingPlayer: int, assisingCard: Card,
inputs: List[Pair[Resource, GridPosition]],
outputs: List[Pair[Resource, GridPosition]],
pollution: List[GridPosition]): bool
SelectReward
player: Optional[int]
selelction: List[Resource]
setReward(player: int, card: Card, reward: Resource[])
canSelectReward(resource: Resource): bool
selectReward(resource: Resource)
state(): string
2
2..4
«use»
Pile
getCard(index: int): Optional[Card]
takeCard(index: int)
removeLastCard()
state(): string
«Datatype»
Player
ScoringMethod
resources: List[Resource]
pointsPerCombination: Points
calculatedTotal: Optional[Points]
selectThisMethodAndCalculate()
state(): string
2
«use»
«Enumeration»
GameState
TakeCardNoCardDiscarded
TakeCardCardDiscarded
ActivateCard
SelectReward
SelectActivationPattern
SelectScoringMethod
Finish
ActivationPattern
pattern: List[GridPosition]
selected: bool
select()
state(): string
2
1
«use»
«Datatype»
Points
Grid
getCard(coordinate: GridPosition): Optional[Card]
canPutCard(coordinate: GridPosition): bool
putCard(coordinate: GridPosition, card: Card)
canBeActivated(coordinate: GridPosition): bool
setActivated(coordinate: GridPosition)
setActivationPattern(pattern: List[GridPosition])
endTurn()
state()
visibleCards
4
hiddenCards
*
1..9
Card
resources:Resource[]
pollutionSpacesL int
canGetResources(resources: List[Resource]): bool
getResources(resources: List[Resource])
canPutResources(resources: List[Resource]): bool
putResources(resources: List[Resource])
check(input: List[Resource], output: List[Resource], polution:int): bool
checkLower(input: List[Resource], output: List[Resource], polution:int): bool
hasAssistance():bool
state(): string
0..1
upperEffect
0..1
lowerEffect
0..1
«Inerface»
Effect
check(input: List[Resource], output: List[Resource], pollution:int): bool
hasAssistance():bool
state(): string
Requires integration tests.
2..*
TransformationFixed
from: List[Resource]
to: List[Resource]
polution: int
ArbitraryBasic
froum: int
to: List[Resource]
polution: int
...
EfectOr