

Pseudo code of ReefGame (implemented in SmallTalk[®] with Cormas[®] platform)

```
<body package="ReefGame" selector="initDicoCatch">initDicoCatch
"the default gear is 'net'; fishers can catch 20 kg/trip with motor boat and 10 kg/trip without motor"
  self dicoCatch: Dictionary new.
  self dicoCatch at: #net put: Dictionary new.
  (self dicoCatch at: #net) at: #no_motor put: 10.
  (self dicoCatch at: #net) at: #motor put: 20.
```

```
<body package="ReefGame" selector="initDicoFish">initDicoFish
"the default fish ratio in mangrove is 10% Carnivore and 90% Herbivore"
  self dicoFish: Dictionary new.
  self dicoFish at: #mangrove put: Dictionary new.
  (self dicoFish at: #mangrove) at: #fishCarni put: 0.1.
  (self dicoFish at: #mangrove) at: #fishHerbi put: 0.9.
"the default fish ratio in coral is 40% Carnivore and 60% Herbivore "
  self dicoFish at: #coral put: Dictionary new.
  (self dicoFish at: #coral) at: #fishCarni put: 0.4.
  (self dicoFish at: #coral) at: #fishHerbi put: 0.6.
"the default fish ratio in algae is 20% Carnivore and 80% Herbivore"
  self dicoFish at: #algae put: Dictionary new.
  (self dicoFish at: #algae ) at: #fishCarni put: 0.2.
  (self dicoFish at:#algae) at: #fishHerbi put: 0.8.
"the default fish ratio in seagrass is 20% Carnivore and 80% Herbivore "
  self dicoFish at: #seagrass put: Dictionary new.
  (self dicoFish at: #seagrass ) at: #fishCarni put: 0.2.
  (self dicoFish at:#seagrass) at: #fishHerbi put: 0.8.
```

```
<body package="ReefGame" selector="initFishBiomass">initFishBiomass
"This method sets the initial fish biomass for each habitat. The maximum biomass per cell is 50kg. Coral can support 100% of the maximum biomass, Algae 60%, Seagrass 50% and Mangrove 40% only."
  | allCorals allSeagrass allAlgae allMangrove maxPossibleBiomass |
  maxPossibleBiomass := 50.
  allCorals := self theCells select: [:c | c habitat = #coral].
  allCorals do: [:c | c fishCarni: (maxPossibleBiomass*((Cell dicoFish at: #coral) at: #fishCarni)) rounded. c fishHerbi: (maxPossibleBiomass*((Cell dicoFish at: #coral) at: #fishHerbi)) rounded].
  allAlgae := self theCells select: [:c | c habitat = #algae].
  allAlgae do: [:c | c fishCarni: (0.6 * (maxPossibleBiomass*((Cell dicoFish at: #algae) at: #fishCarni))) rounded. c fishHerbi: (0.6 * (maxPossibleBiomass*((Cell dicoFish at: #algae) at: #fishHerbi))) rounded].
  allSeagrass := self theCells select: [:c | c habitat = #seagrass].
  allSeagrass do: [:c | c fishCarni: (0.5 * (maxPossibleBiomass*((Cell dicoFish at: #seagrass) at: #fishCarni))) rounded. c fishHerbi: (0.5 * (maxPossibleBiomass*((Cell dicoFish at: #seagrass) at: #fishHerbi))) rounded].
  allMangrove := self theCells select: [:c | c habitat = #mangrove].
  allMangrove do: [:c | c fishCarni: (0.4 * (maxPossibleBiomass*((Cell dicoFish at: #mangrove) at: #fishCarni))) rounded. c fishHerbi: (0.4 * (maxPossibleBiomass*((Cell dicoFish at: #mangrove) at: #fishHerbi))) rounded].
  self theCells do: [:c | c initFish: c fishCarni + c fishHerbi].
  Cell initFishBiomass: ( self seas inject: 0 into: [:tot :x | tot + x fishCarni + x fishHerbi]).
```

```
<body package="ReefGame" selector="step:">step: t
"This is the scheduler of ReefGame that controls the timing and actions of the Fishers, and responses from Fishes and Habitats"
  self updateAquaculture.
  self moveJackpotFish.
  self updatePlayerPositionValues.
  self moveFisher.
  self theFishers do: [:f | f getFish].
```

```

self getFishersExtraIncome.
self theCells do: [:f | f updateHabitat].
self updateHabitatIndex.
self theCells do: [:f | f reproduceFish].
self givePlayerResults.
self theCells do: [:c | c defineVisualState; show]

```

```

<body package="ReefGame" selector="reproduceFish">reproduceFish
"This method allows fish stocks to recover. Population growth coefficients are 1.12 for Coral habitat, 1.07 for Seagrass and Mangrove habitats and 1.05 for Algae habitat"
|newStock|
self habitat = #coral ifTrue: [newStock := (((self fishCarni + self fishHerbi) * 1.12) min: self
initFish)].
self habitat = #seagrass ifTrue: [newStock := (((self fishCarni + self fishHerbi) * 1.07) min: self
initFish)].
self habitat = #mangrove ifTrue: [newStock := (((self fishCarni + self fishHerbi) * 1.07) min: self
initFish)].
self habitat = #algae ifTrue: [newStock := (((self fishCarni + self fishHerbi) * 1.05) min: self
initFish)].
self fishHerbi: (newStock * ((self class dicoFish at: self habitat) at: #fishHerbi)) rounded.
self fishCarni: (newStock * ((self class dicoFish at: self habitat) at: #fishCarni)) rounded.

```

```

<body package="ReefGame" selector="moveJackpotFish">moveJackpotFish
"This method sets the pelagic fish biomass as a random amount between 20 and 100 kg and distributes them in four random Sea cells. It simulates pelagic stocks that pass through municipal fishing waters."
|jackpotFish allSeas mixtSeas jackpotPlaces|
(self seas select: [:c | c habitat = #none]) do: [:x | x fishCarni: 0].
jackpotPlaces := 4.
jackpotFish := Cormas randomFrom: 20 to: 100.
allSeas := self theCells select: [:c | c state = #sea and: [c habitat = #none]].
mixtSeas := Cormas mixt: allSeas asOrderedCollection.
1 to: jackpotPlaces do: [:x | (mixtSeas at: x) fishCarni: (((mixtSeas at: x) fishCarni + jackpotFish)
/ jackpotPlaces) rounded]</body>

```

```

<body package="ReefGame" selector="updateHabitat">updateHabitat
"This method creates a transition shift in reef habitats, from Coral-dominated to Algae-dominated states, when fish stocks fall below a certain proportion of their original value"
self habitat = #coral ifTrue: [self fishCarni + self fishHerbi <= (0.5 * self initFish) ifTrue: [self
habitat: #algae. self initFish: 35]].

```

```

<body package="ReefGame" selector="updateHabitatIndex">updateHabitatIndex
"This method adjusts fishing capacity according to reef habitat degradation. The shift from Coral-dominated to Algae-dominated reefs reduces Fisher's potential catch"
(self theCells select: [:c | c habitat = #coral]) size <= (0.2 * Cell initialCoral)
ifTrue: [Fisher habitatIndex: 0.2].
(self theCells select: [:c | c habitat = #coral]) size <= (0.4 * Cell initialCoral)
ifTrue: [Fisher habitatIndex: 0.4].
(self theCells select: [:c | c habitat = #coral]) size <= (0.6 * Cell initialCoral)
ifTrue: [Fisher habitatIndex: 0.6. ^ self].
(self theCells select: [:c | c habitat = #coral]) size <= (0.8 * Cell initialCoral)
ifTrue: [Fisher habitatIndex: 0.8].

```

```

<body package="ReefGame" selector="getFish">getFish
"This method calculates the catch for each Fisher based on his equipment-driven fishing pressure. The actual catch depends on the level of depletion of the stock. Carnivore fishes are primarily targeted, up to 70% of the possible catch, Herbivore fishes come as a complement."
|fishingPressure catchWanted|
fishingPressure := ((Cell dicoCatch at: self gear) at: self boat).

```

```

catchWanted := ((self patch fishCarni + self patch fishHerbi) * fishingPressure / (self patch
initFish)) rounded.
self catchCarni: ((catchWanted * 0.7) min: self patch fishCarni) rounded.
self patch fishCarni: self patch fishCarni - self catchCarni.
self catchHerbi: ((catchWanted - self catchCarni) min: self patch fishHerbi) rounded.
self patch fishHerbi: self patch fishHerbi - self catchHerbi.
self totalCatch: self totalCatch + self catchHerbi + self catchCarni.

```

```

<body package="ReefGame" selector="getFishersExtraIncome">getFishersExtraIncome
"different off-fishing activities are assigned different base wages"
self theFishers do: [:c | c income: 0].
self getFishersExtraIncomeFrom: #aquaculture withIncome: 200.
self getFishersExtraIncomeFrom: #farming withIncome: 100.
self getFishersExtraIncomeFrom: #building withIncome: 100.
self getFishersExtraIncomeFrom: #taxi withIncome: 150.
self getFishersExtraIncomeFrom: #ferry withIncome: 150.
self getFishersExtraIncomeFrom: #tourism withIncome: 200.

```

```

<body package="ReefGame" selector="getFishersExtraIncomeFrom:withIncome:">
"Fishers earn extra income from a given off-fishing activity based on: a base wage, the number of cells
dedicated to the activity and the number of fishers participating in the activity"
| allFishers allCells |
allFishers := self theFishers select: [:f | f patch notNil and: [f patch livelihood = anActivity]].
allCells := self theCells select: [:f | f livelihood = anActivity].
allFishers isEmpty ifTrue: [^self].
allFishers := (Cormas mixt: allFishers).
allFishers size <= (allCells size * 2)
ifTrue: [1 to: (allFishers size min: allCells size * 2)do: [:x | (allFishers at: x) income: anIncome]]
ifFalse: [1 to: (allCells size * 2) do: [:x | (allFishers at: x) income: anIncome].
(allCells size * 2) + 1 to: allFishers size do: [:x | (allFishers at: x) income: anIncome / 2]].

```

```

<body package="ReefGame" selector="updateAquaculture">updateAquaculture
"Cells converted to aquaculture have no wild fish stocks and lose their Mangrove status"
| newAqua |
newAqua := self theCells select: [:c | c livelihood = #aquaculture].
newAqua isEmpty ifTrue: [^self].
newAqua do: [:c | c state: #land. c habitat: #none. c fishCarni: 0. c fishHerbi: 0].

```