FIT2099 Preliminary Design Documentation

Michael:

Stegosaur and Allosaur will inherit classes Herbivore and Carnivore respectively, which will then inherit Dinosaur.

Hungry Dinosaurs:

Dinosaur hungers will be implemented using behaviours. When a dinosaur is hungry (food level reaches a certain point) , the HungryBehaviour will be called, the message that a Dinosaur is hungry will be displayed from here. The HungryBehaviour will then call MoveToFoodAction which will make the Dinosaur move to the closest food source. Once next to a food, the NearFoodBehaviour will trigger and call either CarnivoreEatAction or HerbivoreEatAction depending on what the Dinosaur eats. NearFoodBehaviour's method will return an integer that is returned from the EatAction's method. The integer is the food level that is gained from either the food.

A Dinosaur becomes unconscious with the UnconciousBehaviour, which is triggered when the food level of Dinosaur is 0, after 20 turns, the DieAction class's method will be called if the Dinosaur is still unconscious.

Breeding:

When a Dinosaur is well-fed and ready to breed, the BreedBehaviour will trigger. Within the BreedBehaviour the Dinosaur will follow and move towards another Dinosaur of the same species and opposite sex. The BreedAction will then trigger where an Egg of the species type will be produced 10 turns later. When an Egg is ready to hatch, the HatchBehaviour class's method is called and the HatchAction's method occurs.

Baby Dinosaur growing into adult Dinosaur is handled within the Dinosaur class itself. There will be a boolean indicating if the dinosaur is an adult or not and after 30 turns the boolean will turn from false (for baby) to true (for adult).

Allosaurs:

When an Allosaur is near a Stegosaur, the AttackBehaviour's method is called and an AttackAction objected is created and its method called.

A dead Dinosaur will produce a DinosaurCorpse object (which is a CarnivoreFood) at the spot it died through the DieAction's method. The Allosaur can then eat that. If a DinosaurCorpse is within range of an Allosaur, the NearFoodBehaviour is triggered and it

will move towards the food with the MoveToFoodAction class. When next to the corpse, CarnivoreEatAction is triggered and the Allosaur eats the corpse. Eggs also inherit from the CarnivoreFood class since Allosaurs (and other carnivores if any) can eat it.

Kenny:

Grass:

In the World class more specifically in the While stillRunning loop , it will call tick method on GameMap,inside the tick function in GameMap it will traverse through all possible locations and invoke the tick function inside of instances of Location. Then inside the tick method of Location invoke another tick method, but this time on Ground Class. The tick method in Ground class is the exact place where we need to go; firstly we will check if the ground is an instance of Dirt or Tree or Grass. If it is a Dirt we will check its surroundings since the location is passed as the parameter we have the option to call getExits() to check its surroundings to see if there Dirt we will have some sort of accumulator, to increase a number. Once done we will use math.random to get a number and if it succeeds then we will create a Grass class inside Dirt, and use the location.setGround method to change the instance of the ground of the location.

Dropping Fruits:

Similar process as Grass using ticks, but rather setting a new ground where just adding an instance of a new Fruit onto the location called addItem via location. If the player wants to harvest fruits we use playTurn method to find the current location of the player and see if there are items on the location of the player and if so, then we grab the items and put them onto the action parameter.

Ecopoints and VendingMachine:

For example, if grass is grown from Dirt then ecopoints must be increased, so it would be similar to how it would be done in the Grass section, but if the constructor of Grass would contain the reference of a location, then we use map() method to get a GameMap. Once we have the GameMap reference it should be very easy, just locate the player reference and then add points to the ecopoints variable.

In terms of how it would be done for the others, the majority of them inherit from Action, and in the execute function there is a reference to GameMap that's all we need we do the same thing as what we did above.

Should probably also be mentioned that Grass and VendingMachine will be created inside the application in the FancyGroundFactory parameter, when FancyGroundFactory is initialized. As with other classes that are visible in the map, VendingMachine will inherit from

Ground so that it can be in the map. As with anything with Ground class it will have a tick method, in here we will check if a player is located in the same spot, and it so give them a buyAction where they could buy one thing from the store for a turn. Although the way we're implementing only allows them to either move, stand-still or purchase items. Once they have purchased the item, it will be added to the player inventory via addItem().