Why is initialization SOOO important???

How transparent should initialization be to the user of your class?

ENTITY INITIALIZATION:

Entity Parameter struct: color, pos, radius

Parameter Struct for each type of Entity that inherits from the Entity Parameter struct (or whatever inheritance structure you build)

Place all the param settings in “main” in an initialization file (NAME????) ...

Define the param values using the structs.

Keep in mind, if useful you can write constructors for structs!

It makes sense to me to have default param values for each struct – IS this in the init file or in the params file?

Build a list of (entity,param) pairs, each of which will be added to Arena.

Arena initializes itself by adding entities based on the list. Thus, need Arena::AddEntity( params )

I think it makes sense to add an initialization method to every entity.

* What about randomization?
* What about reset? Consider the difference between a reset and start (again) – do we want to distinguish? Maybe this is really reset and replay ???

SENSOR, MOTION HANDLER, MOTION BEHAVIOR, BATTERY INITIALIZATION ...

* Is this part of entity parameters for those that have these things?

SENSORS: For each sensor, let’s list information ...

Distress

* How does the sensor information affect entity behavior?
  + Robot
  + SuperBot
  + Player
  + Home Base
  + Recharge Station
* Where is that information used to affect entity behavior?
* What information should the sensor have for the entity to appropriately react?

Entity Type

* How does the sensor information affect entity behavior?
  + Robot
  + SuperBot
  + Player
  + Home Base
  + Recharge Station
* Where is that information used to affect entity behavior?
* What information should the sensor have for the entity to appropriately react?

Proximity

* How does the sensor information affect entity behavior?
  + Robot
  + SuperBot
  + Player
  + Home Base
  + Recharge Station
* Where is that information used to affect entity behavior?
* What information should the sensor have for the entity to appropriately react?

Touch

* How does the sensor information affect entity behavior?
  + Robot
  + SuperBot
  + Player
  + Home Base
  + Recharge Station
* Where is that information used to affect entity behavior?
* What information should the sensor have for the entity to appropriately react?

ARENA POPULATING SENSORS

* Distress Call as Observer Pattern
* Entity Type Emitter as Observer Pattern

ENTITY-OBJECT Events ...

* Proximity
* Touch

Scenario 1: Arena sends 2 entities close to each other to all sensors.

Scenario 2: Arena sends processed information to all sensors.

Scenario 3: Arena sends proximity events to all proximity sensors, touch events to all touch sensors, ...

How do these scenarios differ in implementation? Be specific.

Entity-Object Event Checking:

for m in mobile\_entities\_

for e in entities\_

check boundry

check proximity

check touch

OR

for entities in list of entities that are close ...