

TEALS MINECRAFT PROJECT

Lecture 9: Collector Robot

OVERVIEW

For this lab, we'll create a CollectorBot. This robot entity will

1. Seeks out loose items until the robot comes within range of nearby items. Navigate to the nearest item until close enough to grab it.
2. Picks up the item and adds it to the robot's inventory.
3. On damage, drops all items in its inventory and wanders around stunned for twenty seconds.

1. SEEKS OUT LOOSE ITEMS

This involves enumerating all items in the world, and selecting the closest one within the robot's search distance.

To get a `List<Entity>` of all entities in the world

```
List<Entity> entityList =  
    (List<Entity>) entity.worldObj.getLoadedEntityList();
```

Finding Items Within Range

```
float distance = entity1.getDistanceToEntity(entity2);
```

Navigating to Nearby Items

```
// Start navigation (speed can be Robot.SPEED_FAST, Robot.SPEED_NORMAL,  
// or Robot.SPEED_SLOW).  
entity.getNavigator().tryMoveToEntityLiving(Entity other, float speed);  
  
// Stop navigation.  
entity.getNavigator().clearPathEntity();
```

2. PICK UP LOOSE ITEMS

We will need to keep track of everything that the robot has picked up. We'll also need to "pick up" an item from the world, which means that it's removed from the world.

Removing an item from the world

```
void item.worldObj.removeEntity (EntityItem item);
```

Items are contained in a *stack*, which holds a multiple of some item

```
ItemStack itemStack = entityItem.getEntityItem();  
int count = itemStack.stackSize;  
Item item = itemStack.getItem();
```

Printing items names for debugging

```
String printableItemName = item.getUnlocalizedName();
```

Managing the inventory of collected items — how?

HASHMAPS

You know how arrays work. An array contains a set of values, which you reference with an integer:

```
int[] someArray;  
someArray[0] = 37;  
someArray[1] = 13;  
someArray[2] = 11;  
int foo = someArray[1];    // foo ← 13
```

What if you could index an array with any value?

```
HashMap<String,Integer> hash = new HashMap<String,Integer>();  
hash.put ("Ariel", 37);           // hash["Ariel"] ← 37  
hash.put ("Bubbles", 13);        // hash["Bubbles"] ← 13  
hash.put ("Calisto", 11);        // hash["Calisto"] ← 11  
int foo = hash.get ("Bubbles");  // foo ← hash["Bubbles"]  
hash.remove ("Ariel");           // Delete hash["Ariel"]
```

That's a hash map.

HASHMAPS

```
HashMap<String,Integer> hash = new HashMap<String,Integer>();
hash.put ("Ariel", 37);           // hash["Ariel"] ← 37
hash.put ("Bubbles", 13);        // hash["Bubbles"] ← 13
hash.put ("Calisto", 11);        // hash["Calisto"] ← 11
int foo = hash.get ("Bubbles");  // foo ← hash["Bubbles"]
hash.remove ("Ariel");           // Delete hash["Ariel"]
```

Enumerating HashMaps

```
Set<KeyType> keyset = hash.keySet();    // Get set of all hash keys

// Print the contents of a hash map
for (KeyType key : hash.keySet()) {
    valueType value = hash.get(key);
}

// or

for (Item item : inventory.keySet()) {
    Integer count = inventory.get(item);
}
```

3. ON DAMAGE, STUNNED ROBOT DROPS ITEMS

When the robot experiences damage of any kind, it drops all of its items and wanders around stunned for twenty seconds.

Handle Robot Damage Events

```
// This Robot method will be called whenever our collector bot  
receives // any kind of damage.  
public void Robot.onEntityDamage (DamageSource source, float amount);
```

Drop All Items In the Robot's Inventory

```
// Do this for each item in the robot's inventory  
void dropItem (Item item, int count);
```

3. ON DAMAGE, STUNNED ROBOT DROPS ITEMS

The final piece of wandering around stunned for 20 seconds is that we need to suppress the robot's item-collection task.

Remember these EntityAIBase methods?

```
// Indicate whether the task (collection task in this case) should execute  
public boolean shouldExecute();  
  
// Indicate whether the task (collection task) should continue executing  
public boolean continueExecuting();
```

When we're stunned (when the robot is damaged in any way), then `continueExecuting()` will return false. Using the System time, we'll always return false from `shouldExecute()` as long as we're in "stun time".

LAB 9: COLLECTOR ROBOT

At this point you should have everything you need to implement the collector robot. Remember to reference the notes section, as this will have all of the details you need to complete this lab.