

CSE 230 Problem Set 12

Problem 28.1: Position Attributes

Consider the following problem definition:

An orbital simulator has a notion of a position. This represents the 2-dimensional location of an orbital body in the simulation. A position is stored, can be copied from one position to another, and can be moved according to a velocity. Note that you can represent the position in meters or in pixels.

Using appropriate metaphors, create a class diagram describing a class to satisfy this problem.

Position
<ul style="list-style-type: none">- X: double- Y: double
<ul style="list-style-type: none">+ Position()+ Position(x: double, y: double)+ Operator=(pos: Position): Position+ SetX(x: double)+ SetY(y: double)+ GetXMeters: double+ GetYMeters: double+ GetXPixels: double+ GetYPixels: double+ Move(velocity, angle)

Problem 28.2: Orbital Element Collection

Consider the following problem definition:

An orbital simulator keeps track of many elements (such as satellites, planets, etc), each of which responds to the force of gravity. This **collection is reduced** as elements fall to earth or **increase when** new items are added to the simulation.

Using appropriate metaphors, create a class diagram to satisfy this problem:

Elements
- Collection :[Object]
+ Remove(item: Object) + Insert(item: Object) + OnFall(Remove())

Problem 28.3: User Interface

Consider the following problem definition:

An orbital simulator allows the user to control various elements in the simulation. A satellite, for example, can thrust to a higher orbit. Also, the user can add new elements to the simulation. All the while, the simulator displays the current status to the user.

Using appropriate metaphors, create a class diagram to satisfy this problem:

UserInterface	
-	Status
-	Elements[]
+	DisplayStatus()
+	PromptElements()
+	PromptElementControl()
+	InputElement()

Problem 28.4: Velocity Operations

Consider the following problem definition:

An orbital simulator has a notion of velocity. A velocity the speed and direction (from Example 28.3) of movement. It is possible to increase velocity by a fixed amount (such as add 5 m/s to the speed), increase velocity by a factor (such as double the speed), reverse velocity (move the opposite direction), and copy a velocity object. It is also possible to combine two velocities, equivalent to a nudge given to a marble rolling down a slope.

Using appropriate metaphors, create a class diagram to satisfy this problem:

Velocity
<ul style="list-style-type: none">- Speed- Direction
<ul style="list-style-type: none">+ Operator+(v: Velocity): Velocity+ Operator+=(amount: double): Velocity+ Operator*=(factor: double) Velocity+ Operator=(v: Velocity): Velocity+ Reverse(v: Velocity): Velocity

Problem 28.5: Orbital Element

Consider the following problem definition:

An orbital simulator models the interactions of various orbital elements. These elements are part of a larger collection of elements (described in Problem 28.2). Each individual element has a position (Problem 28.1), velocity (Problem 28.4), and has a variety of status attributes (such as whether it is dead and needs to be removed from the simulation), and can handle a collection of events (when two elements collide, when it is time to draw, when it is time to move). Using appropriate metaphors, describe a function to handle such events.

Using appropriate metaphors, create a class diagram to satisfy this problem:

OrbitalElement
<ul style="list-style-type: none">- Position- Velocity- Status
<ul style="list-style-type: none">+ OnCollision()+ OnDraw()+ OnMove()+ OnDeath()+ SetStatus()