6.905/6.945 Final Project Draft: Extensible Physics Engine

Due on 8 May 2017

Author1, Author2, Author3

1   Introduction

blah blah blah

2   Infrastructure

generic procedures, ps4 stuff what we did, how we did it, why we did it

3   Arithmetic

We incorporated the generic arithmetic developed in pset 3. We extended the vector-arithmetic to support element-wise operations between scalars and vectors. We did this by coercing scalars to become vectors that contain the same value. For example, `(+ #(3 4) 1)` gets transformed into `(+ #(3 4) (1 1))`. The coercian method is

```
(define (coerce a size-reference)
  (if (vector? a)
    a
    (make-vector (vector-length size-reference) a)))
```

The rest of the infrastructure for generic procedures was adapted from pset 4, but pset 4's generic arithmetic is not compatible with that of pset 3. To fix this issue, we first loaded the relevant pset 3 dependencies, installed a generic arithmetic with vector operations, redefined arithmetic operations to be the ones from that generic arithmetic (such as with `(define * (access * arith-environment))`), and then loaded the pset 4 generic procedure infrastructure on top.

4   Physics Engine

each object does its own updating, procedures are objects what we did, how we did it, why we did it

5 Demos

5.1 Gravity

We can easily extend our system with new forces and new objects. We begin by creating an gravity as an interaction:

```
(define (make-gravity thing all-things)
  (define (procedure thing influences)
    (sum (map (lambda (influence)
                (let* ((m1 (get-mass thing))
                       (m2 (get-mass influence))
                       (G 6.674e-11)
                       (v (- (get-position influence)  ; vector between influence and thing
                             (get-position thing)))
                       (r (magnitude v))                ; distance between influence and thing
                       (u (/ v r))                      ; unit vector
                       (gmag (* (* G m1 m2)             ; magnitude of gravity
                                (/ 1 (square r)))))
                  (* u gmag)))                          ; gravitational force vector
              influences)))
  (let ((influences (delq thing all-things)))
    (make-interaction gravity? 'gravity procedure influences)))
```
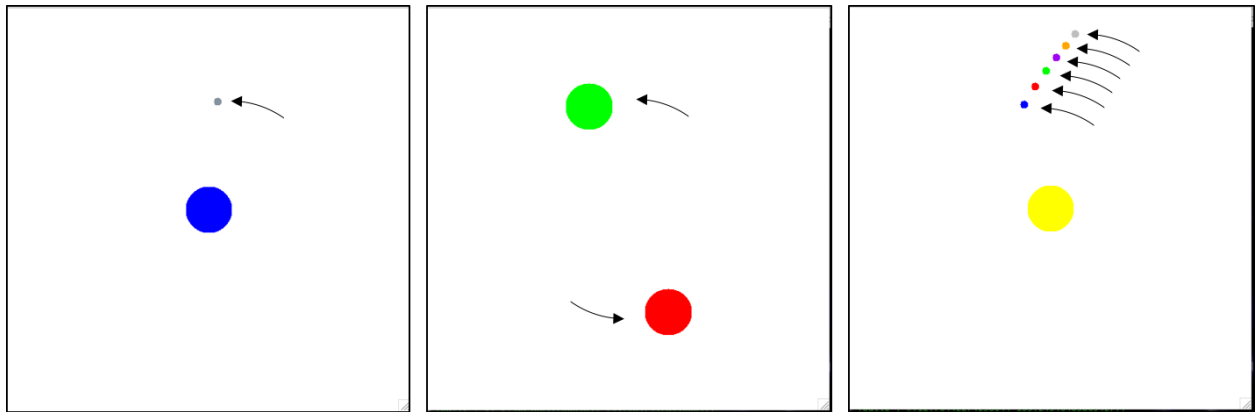

Figure 1: [*left*] `earth-moon`; [*center*] `binary-stars`; [*right*] `solar-system`

We've created several worlds that exhibit gravity (Fig. 1). Once an object is added to the world, any object that has mass will feel gravitational forces with other object that have mass.

This world contains an earth with a moon orbiting around it.

```
(define (earth-moon)
  (define w (make-world "world"))
  (define b1 (make-ball "earth" 30 1e15 #(0 0) #(0 0) "blue"))
```

```
  (define b2 (make-ball "moon" 5 1e5 #(100 100) #(-15.361 15.361) "#85929E"))

  (add-mass! b1 w)
  (add-mass! b2 w)
  w)
```

This world contains two equal masses orbiting a common center of mass.

```
(define (binary-stars)
  (define w (make-world "world"))
  (define b1 (make-ball "ball1" 5 1e15 #(-100 -100) #(9 -9) "red"))
  (define b2 (make-ball "ball2" 5 1e15 #(100 100) #(-9 9) "green"))

  (add-mass! b1 w)
  (add-mass! b2 w)
  w)
```

This world contains a massive sun and several planets orbitting around the sun.

```
(define (solar-system)
  (define w (make-world "world"))
  (define s (make-ball "sun" 30 1e15 #(0 0) #(0 0) "yellow"))
  (define b1 (make-ball "ball1" 5 1e5 #(100 100) #(-15.361 15.361) "blue"))
  (define b2 (make-ball "ball2" 5 1e5 #(110 110) #(-15.361 15.361) "red"))
  (define b3 (make-ball "ball3" 5 1e5 #(120 120) #(-15.361 15.361) "green"))
  (define b4 (make-ball "ball4" 5 1e5 #(130 130) #(-15.361 15.361) "purple"))
  (define b5 (make-ball "ball5" 5 1e5 #(140 140) #(-15.361 15.361) "orange"))
  (define b6 (make-ball "ball6" 5 1e5 #(150 150) #(-15.361 15.361) "gray"))

  (add-mass! s w)
  (add-mass! b1 w)
  (add-mass! b2 w)
  (add-mass! b3 w)
  (add-mass! b4 w)
  (add-mass! b5 w)
  (add-mass! b6 w)
  w)
```

5.2   Magentism

Let's add magnetism into the physics engine.

```
(define (make-magnetic-force magnet all-magnets)
  (define (procedure magnet influences)
    (sum (map (lambda (influence)
                (let* ((q1 (get-magnet-charge magnet))
                       (q2 (get-magnet-charge influence))
                       (mu 1.256e-6)  ; permeability of air
                       (v (- (get-position influence)  ; vector between influence and magnet
                         (get-position magnet)))
```

```
                    (r (magnitude v))                  ; distance between influence and magnet
                    (u (* -1 (/ v r)))                 ; unit vector
                    (mmag (* (* mu q1 q2)              ; magnitude of magnetic force
                          (/ 1 (* 4 pi (square r))))))))
                (* u mmag)))                           ; magnetic force vector
            influences)))
  (let ((influences (delq magnet all-magnets)))
    (make-interaction magnetic-force? 'magnetic-force procedure influences)))
```

This is a simple world that two magnets repelling each other.

```
(define (magnets-1)
  (define w (make-world "world"))
  (define m1 (make-magnet "magnet1" 1e5 1 #(-30 -30) #(0 0) "red"))
  (define m2 (make-magnet "magnet2" 1e5 1 #(30 30) #(0 0) "red"))

  (add-magnet! m1 w)
  (add-magnet! m2 w)
  w)
```

This world contains two positively charged masses (blue) and two negatively charged masses (red).

```
(define (magnets-2)
  (define w (make-world "world"))
  (define m1 (make-magnet "magnet1" 5e5 1 #(-100 -100) #(0 0) "red"))
  (define m2 (make-magnet "magnet2" -5e5 1 #(100 100) #(0 0) "blue"))
  (define m3 (make-magnet "magnet3" 5e5 1 #(-100 100) #(0 0) "red"))
  (define m4 (make-magnet "magnet4" -5e5 1 #(100 -100) #(0 0) "blue"))

  (add-magnet! m1 w)
  (add-magnet! m2 w)
  (add-magnet! m3 w)
  (add-magnet! m4 w)
  w)
```

We can also combine gravity and magnetism together. Here, we created a solar-system-like world where each of the planets repel each other and the sun attracts the planets via gravity and magnetism. As figure 2 (right) shows, this magnetic solar system exhibits wider and diverging orbits compared to figure 1 (right), the solar system with only gravity.

```
(define (magnetic-solar-system)
  (define w (make-world "world"))
  (define s (make-magnet "sun" 5e5 1e15 #(0 0) #(0 0) "blue"))
  (define b1 (make-magnet "ball1" -2e7 1e5 #(100 100) #(-15.361 15.361) "red"))
  (define b2 (make-magnet "ball2" -2e7 1e5 #(110 110) #(-15.361 15.361) "red"))
  (define b3 (make-magnet "ball3" -2e7 1e5 #(120 120) #(-15.361 15.361) "red"))
  (define b4 (make-magnet "ball4" -2e7 1e5 #(130 130) #(-15.361 15.361) "red"))
  (define b5 (make-magnet "ball5" -2e7 1e5 #(140 140) #(-15.361 15.361) "red"))
  (define b6 (make-magnet "ball6" -2e7 1e5 #(150 150) #(-15.361 15.361) "red"))

  (add-magnet! s w)
  (add-magnet! b1 w)
```

```
(add-magnet! b2 w)
(add-magnet! b3 w)
(add-magnet! b4 w)
(add-magnet! b5 w)
(add-magnet! b6 w)
w)
```
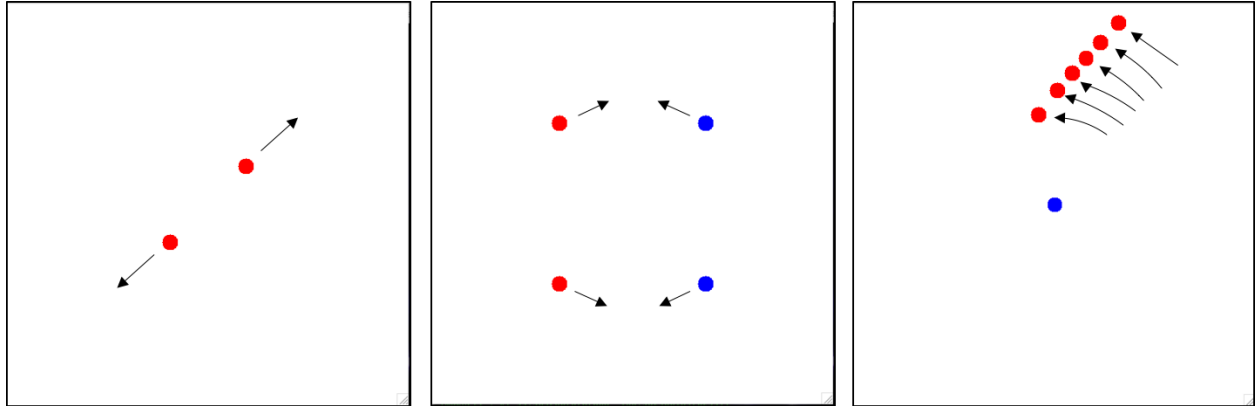

Figure 2: [*left*] `magnets-1`; [*center*] `magnets-2`; [*right*] `magnetic-solar-system`