1 Tower Defense

1.1 Additional Preliminaries

Save your solutions for this problem to a file named defense.rkt.

Run the following expression (you must have required [extras.rkt](http://www.ccs.neu.edu/course/cs5010sp15/files/extras.rkt)) to check that your file is properly named and is in the proper directory:

(check-location "13" "defense.rkt")

Add these additional provides at the top of your file (below the requires), so that we can test your solution:

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) Unit<%>)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) StatefulWorld<%>)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) mk-world)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) mk-ally)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) mk-enemy)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) mk-merc)

1.2 Problem Description

The main objective of this assignment is to create a small "tower defense" game using OOP.

**Game:**

There’s a small war going on and you are on top of a tower with a cross bow, guarding a base. You can see 3 types of units advancing towards you:

* AllyUnit: These are your reinforcements arriving to help. Represent them as a solid green [square](http://docs.racket-lang.org/teachpack/2htdpimage.html#%28def._%28%28lib._2htdp%2Fimage..rkt%29._square%29%29) with a side of 20*pixels*.
* EnemyUnit: These are enemies attacking the base. Represent them as solid red [circle](http://docs.racket-lang.org/teachpack/2htdpimage.html" \l "%28def._%28%28lib._2htdp%2Fimage..rkt%29._circle%29%29)with a radius of 12*pixels*.
* MercenaryUnit: These units switch sides from enemy to ally every 3*ticks*.
  + When the game begins they behave and look like an AllyUnit.
  + On the third tick, they turn into an EnemyUnit.
  + This cycle repeats for the remainder of the game. Specifically, imagine there exists atick-merc function, an INITIAL-MERC constant, and ally? and enemy? predicates. Then the following tests should pass. **NOTE:** This code is meant only to demonstrate the timing of the transitions. It may or may not be a good idea to write these actual functions.

|  |
| --- |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) ally? INITIAL-MERC) |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) ally? (tick-merc INITIAL-MERC)) |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) ally? (tick-merc (tick-merc INITIAL-MERC))) |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) enemy? (tick-merc (tick-merc (tick-merc INITIAL-MERC)))) |

The crossbow should be represented as a target that mirrors the movement of the mouse. Clicking on a unit such that the center of the target is within the bounds of the unit’s shape removes that unit from the game.

**Objective:**

Eliminate EnemyUnits by clicking on them and allow AllyUnits to enter the base.

A unit "enters" the base if there is overlap between the unit’s shape and the base or if their edges touch.

For MercenaryUnits:

* If at the time of entering the base if this unit is an enemy then you will lose points.
* If at the time of entering the base if this is an ally then you will gain points.

**Scoring:**

* EnemyUnit eliminated: ( +40 )
* EnemyUnit reaches Base: ( -40 )
* AllyUnit eliminated: ( -20 )
* AllyUnit reaches Base: ( +20 )
* MercenaryUnit eliminated:
  + when it is impersonating an EnemyUnit: ( +60 )
  + when it is impersonating an AllyUnit: ( -60 )
* MercenaryUnit reaches Base:
  + when it is impersonating an EnemyUnit: ( -60 )
  + when it is impersonating an AllyUnit: ( +60 )

**Displaying the game:**

* The World is displayed as a canvas of *width =*400 and *height =*500*pixels*.
* The Base is a yellow solid [rectangle](http://docs.racket-lang.org/teachpack/2htdpimage.html#%28def._%28%28lib._2htdp%2Fimage..rkt%29._rectangle%29%29) of *width =*400 and *initial height =*50*pixels* and situated at the bottom of the canvas.
* The score should be displayed in the middle of the base and should fit within the bounds of the base.

**Other game rules:**

* The height of the base also represents your progress in the battle and should grow or shrink as the score increases or decreases such that *height of base =*50*+ score/*5*pixels*.
* The game ends when either
  + *height of base <=*10*pixels*, i.e., *score <= -*200
  + *height of base >=*500*pixels*, i.e., *score >=*2250
* The aiming point is represented as two concentric [circle](http://docs.racket-lang.org/teachpack/2htdpimage.html#%28def._%28%28lib._2htdp%2Fimage..rkt%29._circle%29%29)s and a "plus sign" crosshair. The inner and outer radii should be 5 and 10*pixels*, respectively. The crosshair should be a vertical "plus sign" of height and width equal to the diameter of the outer circle.
* As soon as an unit reaches the base it should be removed from the game state. A unit "reaches" the base when either their edges touch or any part of the unit overlaps with the base.
* The game starts with some number of units at the top of the canvas. A new random unit should enter from the top of the canvas every 4*ticks*. Entering units should have a random velocity between a specified min and max velocity, a center x coordinate within the canvas bounds, and a center y coordinate of 0.

**Interfaces and functions to provide:**

|  |
| --- |
| ; Represents a unit in the game. |
| (define Unit<%> |
| ([interface](http://docs.racket-lang.org/reference/createinterface.html#%28form._%28%28lib._racket%2Fprivate%2Fclass-internal..rkt%29._interface%29%29) () |
| ; get-loc : -> [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29) |
| ; Returns the location of this unit as a [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29). |
| get-loc |
|  |
| ; get-color : -> [Color](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._color%29) |
| ; Returns the color of this unit. |
| get-color)) |
|  |
| ; Represents a mutable world object. |
| (define StatefulWorld<%> |
| ([interface](http://docs.racket-lang.org/reference/createinterface.html#%28form._%28%28lib._racket%2Fprivate%2Fclass-internal..rkt%29._interface%29%29) () |
| ; on-tick! : -> [Void](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._void%29) |
| ; EFFECT: mutates this world to its next state. |
| on-tick! |
|  |
| ; on-mouse! : [Coordinate](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._coordinate%29) [Coordinate](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._coordinate%29) [MouseEvent](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._mouseevent%29) -> [Void](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._void%29) |
| ; EFFECT: mutates this world to its next state from the given mouse parameters. |
| on-mouse! |
|  |
| ; target-loc : -> [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29) |
| ; Returns the center of the target as a [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29). |
| target-loc |
|  |
| ; get-units : -> ListOf<Unit<%>> |
| ; Returns the current units in this world. |
| get-units |
|  |
| ; add-unit! : Unit<%> -> [Void](http://www.ccs.neu.edu/course/cs5010sp15/predatadefs.html#%28tech._void%29) |
| ; EFFECT: adds the given unit to the world |
| add-unit! |
|  |
| ; get-base-height : -> Natural |
| ; Returns the height of the base, in pixels. |
| get-base-height)) |
|  |
| ; A Velocity is a Natural, representing Pixels/tick in the downward direction. |
|  |
| ; mk-world : Velocity Velocity Natural -> StatefulWorld<%> |
| ; Creates a world with num-units initial random units, |
| ; where units have the specified min and max velocity. |
| ; WHERE: minvel <= maxvel |
| (define (mk-world maxvel minvel num-units) ...) |
|  |
| ; mk-enemy : [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29) Velocity -> Unit<%> |
| ; Creates an enemy unit with the given parameters. |
|  |
| ; mk-ally : [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29) Velocity -> Unit<%> |
| ; Creates an ally unit with the given parameters. |
|  |
| ; mk-merc : [posn](http://docs.racket-lang.org/htdp/index.html#%28def._%28%28lib._lang%2Fposn..rkt%29._posn%29%29) Velocity -> Unit<%> |
| ; Creates a mercenary unit with the given parameters. |