Assignment: Racket Functions and Recursion

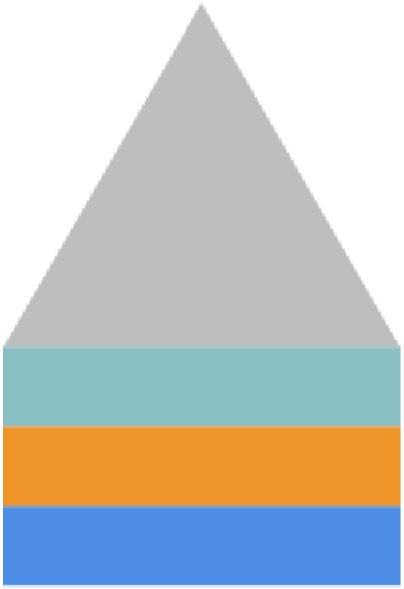
Abstract: This assignment had various tasks that asked me to solve problems. The way the problems had to be solved was with the restriction of using recursion only. The purpose of this assignment is to get me familiar with thinking recursive thoughts and become more familiar with programming.

Interaction: Colorful Permutations of Tract Houses

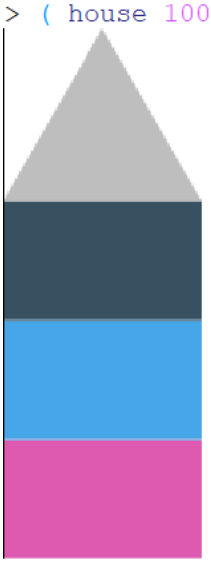
Demo:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( house 200 40 ( random-color ) ( random-color ) ( random-color ) )
```



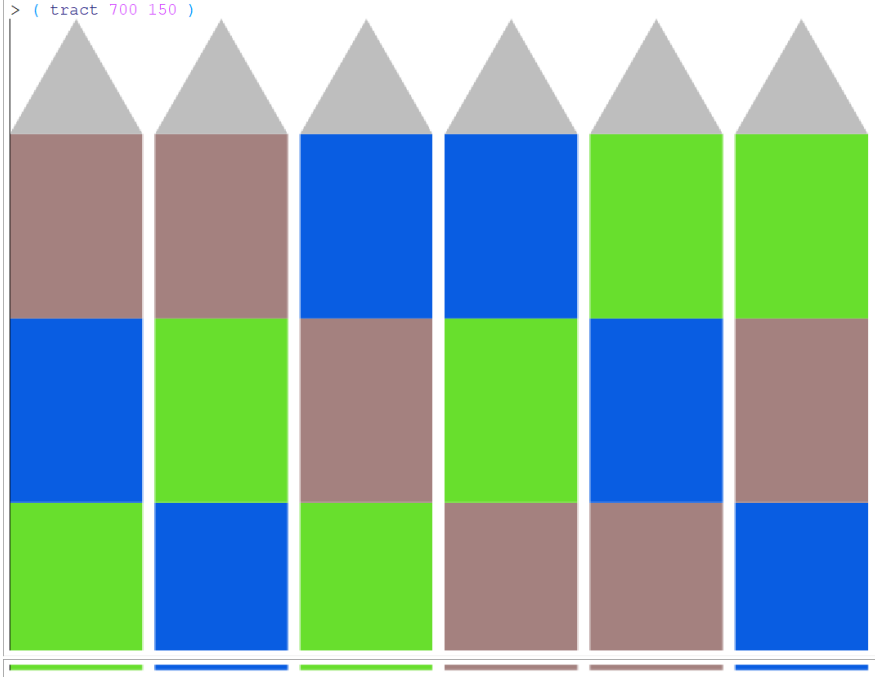
```
> ( house 100 60 ( random-color ) ( random-color ) ( random-color ) )
```



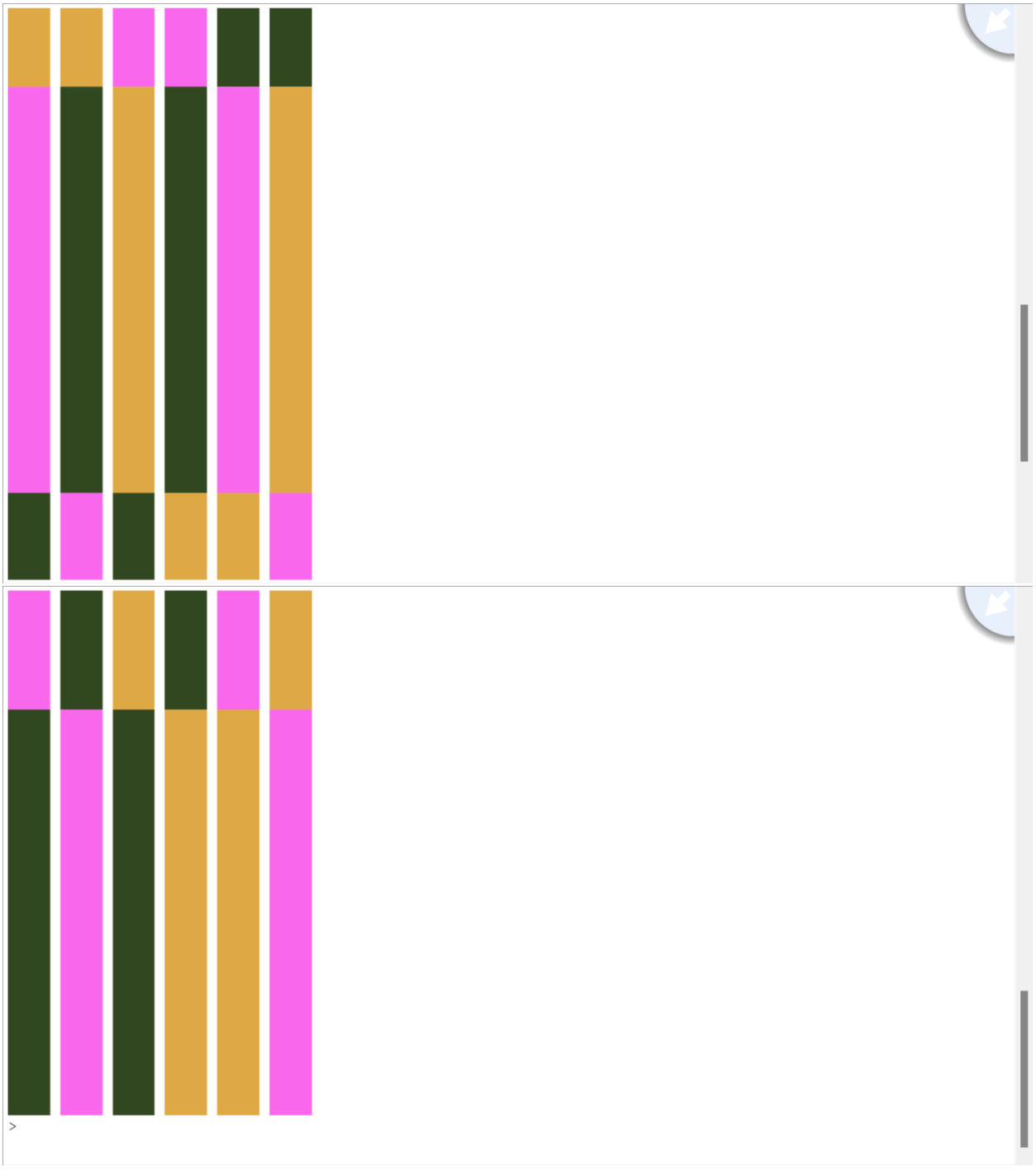
```
>
```

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( tract 700 150 )
```



```
> ( tract 300 400 )
```



Code:

```
( require 2htdp/image )

( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )

( define ( rgb-value ) ( random 256 ) )
```

```
( define ( house width height first-color second-color third-color )
   ( define first-floor ( rectangle width height "solid" first-color )
)
   ( define second-floor ( rectangle width height "solid" second-color
) )
   ( define third-floor ( rectangle width height "solid" third-color )
)
   ( define roof ( triangle width "solid" "gray" ) )
   ( define the-house ( above roof third-floor second-floor first-
floor ) )
   the-house
   )
( define ( tract width height )
   ( define width-of-house ( / ( - width 50 ) 6 ) )
   ( define space ( rectangle 10 0 "solid" "white" ) )
   ( define color1 ( random-color ) )
   ( define color2 ( random-color ) )
   ( define color3 ( random-color ) )
   ( define house1 ( house width-of-house height color3 color2 color1
) )
   ( define house2 ( house width-of-house height color2 color3 color1
) )
   ( define house3 ( house width-of-house height color3 color1 color2
) )
   ( define house4 ( house width-of-house height color1 color3 color2
) )
   ( define house5 ( house width-of-house height color1 color2 color3
) )
   ( define house6 ( house width-of-house height color2 color1 color3
) )
   ( define the-tract ( beside house1 space house2 space house3 space
house4 space house5 space house6 ) )
   the-tract
   )
```

Interaction: Dice

Demo:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( roll-die )
3
> ( roll-die )
6
> ( roll-die )
1
> ( roll-die )
3
> ( roll-die )
6
> ( roll-for-1 )
3 3 3 5 6 5 4 6 6 1
> ( roll-for-1 )
1
> ( roll-for-1 )
1
> ( roll-for-1 )
6 5 2 1
> ( roll-for-1 )
1
> ( roll-for-odd-even-odd )
5 2 5
> ( roll-for-odd-even-odd )
1 6 2 6 1 1 3 6 2 5 1 1 4 4 5 3 6 6 2 6 2 6 5 2 6 1 1 5 4 6 4 4 1 5 3 5 6 6 2
4 3 2 6 2 3 4 3
> ( roll-for-odd-even-odd )
2 5 3 6 4 6 5 2 1
> ( roll-for-odd-even-odd )
3 5 4 3 4 1
> ( roll-for-odd-even-odd )
3 2 4 1 2 4 4 4 3 3 6 6 1 2 3
> ( roll-two-dice-for-lucky-pair )
(1 2) (3 6) (4 4)
> ( roll-two-dice-for-lucky-pair )
```

```
6 5 2 1
> ( roll-for-1 )
1
> ( roll-for-odd-even-odd )
5 2 5
> ( roll-for-odd-even-odd )
1 6 2 6 1 1 3 6 2 5 1 1 4 4 5 3 6 6 2 6 2 6 5 2 6 1 1 5 4 6 4 4 1 5 3 5 6 6
4 3 2 6 2 3 4 3
> ( roll-for-odd-even-odd )
2 5 3 6 4 6 5 2 1
> ( roll-for-odd-even-odd )
3 5 4 3 4 1
> ( roll-for-odd-even-odd )
3 2 4 1 2 4 4 4 3 3 6 6 1 2 3
> ( roll-two-dice-for-lucky-pair )
(1 2) (3 6) (4 4)
> ( roll-two-dice-for-lucky-pair )
(2 4) (3 3)
> ( roll-two-dice-for-lucky-pair )
(6 4) (5 3) (3 6) (1 5) (4 4)
> ( roll-two-dice-for-lucky-pair )
(1 3) (2 2)
> ( roll-two-dice-for-lucky-pair )
(4 2) (6 5)
> ( roll-two-dice-for-lucky-pair )
(4 1) (1 5) (6 3) (5 6)
> ( roll-two-dice-for-lucky-pair )
(6 1)
> ( roll-two-dice-for-lucky-pair )
(6 2) (6 5)
> ( roll-two-dice-for-lucky-pair )
(5 4) (1 3) (5 1) (2 1) (1 2) (5 5)
> ( roll-two-dice-for-lucky-pair )
(1 5) (3 3)
>
```

Code:

```
#lang racket

( define ( roll-die ) ( + ( random 6 )  1 ) )

( define ( roll r )

   ( cond

      ( ( not ( zero? r ) )

        ( display ( roll-die ) ) ( display " " )

        ( roll ( - r 1 ) )

        )

      )

   )
```

```
( define ( roll-for-1 )
   ( define result ( roll-die ) )
   ( display result ) ( display " " )
   ( cond
      ( ( not ( eq? result 1 ) )
        ( roll-for-1 )
        )
      )
   )
( define ( roll-for-11 )
   ( roll-for-1 )
   ( define result ( roll-die ) )
   ( display result ) ( display " " )
   ( cond
      ( ( not ( eq? result 1 ) )
        ( roll-for-11 )
        )
      )
   )
( define ( roll-for-odd )
   ( define result ( roll-die ) )
   ( display result ) ( display " " )
   ( cond
      ( ( = ( remainder result 2 ) 0 )
        ( roll-for-odd )
        )
      )
   )

( define ( roll-for-odd-even )
```

```
   ( roll-for-odd )
   ( define result ( roll-die ) )
   ( display result ) ( display " " )
   ( cond
      ( ( > ( remainder result 2 ) 0 )
        ( roll-for-odd-even )
        )
      )
   )
( define ( roll-for-odd-even-odd )
   ( roll-for-odd-even )
   ( define result ( roll-die ) )
   ( display result ) ( display " " )
   ( cond
      ( ( = ( remainder result 2 ) 0 )
        ( roll-for-odd-even-odd )
        )
      )
   )
( define ( roll-two-dice-for-lucky-pair )
   ( define result1 ( roll-die ) )
   ( define result2 ( roll-die ) )
   ( display "(" ) (display result1 ) ( display " " ) ( display
result2 ) ( display ")" ) ( display " " )
   ( cond
      ( ( not ( eq? ( + result1 result2 ) 7 ) )
        ( cond
           ( ( not ( eq? ( + result1 result2 ) 11 ) )
             ( cond
                ( ( not ( eq? result1 result2 ) )
                  ( roll-two-dice-for-lucky-pair )
```

```
                    )
                  )
                )
              )
            )
          )
        )
```

Interaction: Number Sequences

Demo:

```
> ( triangular 1 )
1
> ( triangular 2 )
3
> ( triangular 3 )
6
> ( triangular 4 )
10
> ( triangular 5 )
15
> ( sequence triangular 20 )
1 3 6 10 15 21 28 36 45 55 66 78 91
105 120 136 153 171 190 210
```

Demo:

```
>  ( square 5 )
25
> ( square 10 )
100
> ( sequence square 15 )
1 4 9 16 25 36 49 64 81 100 121 144
169 196 225
> ( cube 2 )
8
> ( cube 3 )
27
> ( sequence cube 15 )
1 8 27 64 125 216 343 512 729 1000
1331 1728 2197 2744 3375
```

Demo:

```
> ( sigma 1 )
1
> ( sigma 2 )
3
> ( sigma 3 )
4
> ( sigma 4 )
7
> ( sigma 5 )
6
> ( sequence sigma 20 )
1 3 4 7 6 12 8 15 13 18 12 28 14 24 24
31 18 39 20 42
>
```

Code:

```
#lang racket

( define ( square n )
   ( * n n )
   )


( define ( cube n )
   ( * n n n )
   )



( define ( sequence name n )
   ( cond
      ( ( = n 1 )
        ( display ( name 1 ) ) ( display " " )
        )
      ( else
        ( sequence name ( - n 1 ) )
        ( display ( name n ) ) ( display " " )
        )
      )
   )
```

```
( define ( triangular n )
   ( cond
      ( ( = n 1 )
        n
        )
      ( ( > n 1 )
        ( + n ( triangular ( - n 1 ) ) )
      )
      )
   )
( define ( sigma n )
   ( factors 1 n )
   )

( define ( factors from to )
   ( cond
      ( ( = from to )
        ( factor from to )
        )
      ( ( < from to )
        ( + ( factor from to ) ( factors ( + from 1 ) to ) )
        )
      )
   )
( define ( factor m n )
   ( cond
      ( ( = ( remainder n m ) 0 )
        m
        )
```

```
      ( ( not ( = ( remainder n m ) 0 ) )
        0
        )
      )
  )
```

Interaction: Hirst Dots

Demo:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( hirst-dots 10 )
```

```
> ( hirst-dots 4 )
```

```
>
```

Code:

```
#lang racket
( require 2htdp/image )
( define dot-diameter 30 )
( define dot-radius ( / dot-diameter 2 ) )
( define hspace ( rectangle 10 0 "solid" "white" ) )
```

```
( define vspace ( rectangle 0 10 "solid" "white" ) )

( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-
value ) ) )

( define ( rgb-value ) ( random 256 ) )

( define ( dot )

   ( define radius dot-radius )

   ( define color ( random-color ) )

   ( define the-dot ( circle radius "solid" color ) )

   the-dot

   )


( define ( row-of-dots n )

   ( cond

      ( ( = n 0 )

        ( display "\n" )

        )

      ( ( = n 1 )

        ( dot )

        )

      ( ( > n 1 )

        ( beside ( dot ) hspace ( row-of-dots ( - n 1 ) ) )

        )

      )

   )

( define ( rectangle-of-dots r c )

   ( cond

      ( ( = c 0 )

        vspace

        )

      ( ( = r 0 )

        vspace
```

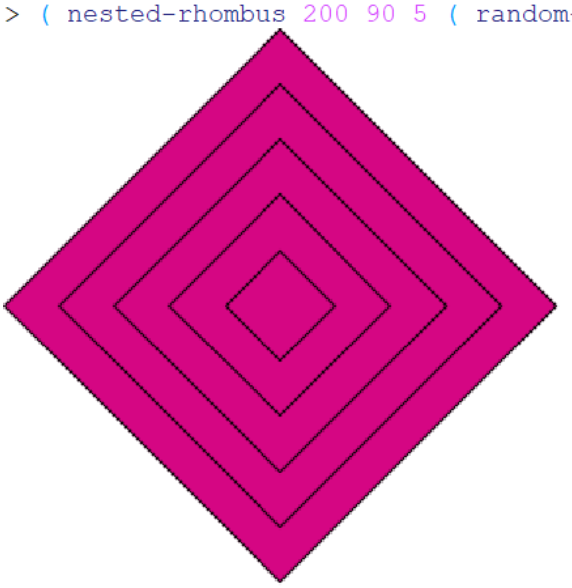
```
        )
      ( ( > r 0 )
        ( above ( row-of-dots c ) vspace ( rectangle-of-dots ( - r 1 )
c ) )
        )
      )
   )
( define ( hirst-dots n )
   ( rectangle-of-dots n n )
   )
```

Interaction: Channeling Frank Stella

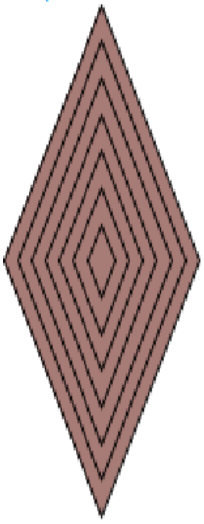
Demo:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( nested-rhombus 200 90 5 ( random-color ) )
```



```
> ( nested-rhombus 140 42 7 ( random-color ) )
```



```
>
```

Code:

```
#lang racket

( require 2htdp/image )

( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )

( define ( rgb-value ) ( random 256 ) )


( define ( nested-rhombus side angle count color )
```

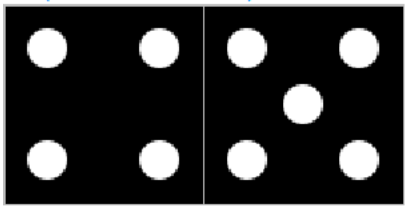
```
   ( define unit ( / side count ) )
   ( paint-nested-rhombus 1 count unit angle color )
   )
( define ( paint-nested-rhombus from to unit angle color )
   ( define side-length ( * from unit ) )
   ( cond
      ( ( = from to )
        ( framed-rhombus side-length angle color )
        )
      ( ( < from to )
        ( overlay
          ( framed-rhombus side-length angle color )
          ( paint-nested-rhombus ( + from 1 ) to unit angle color )
          )
        )
      )
   )
( define ( framed-rhombus side-length angle color )
   ( overlay
     ( rhombus side-length angle "outline" "black" )
     ( rhombus side-length angle "solid" color )
     )
   )
```

Interaction: Dominos

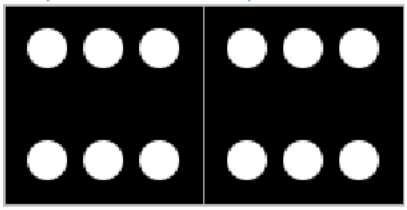
Demo:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

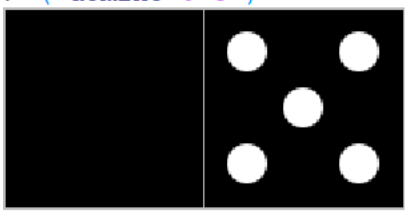
```
> ( domino 4 5 )
```



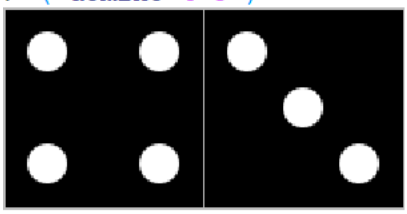
```
> ( domino 6 6 )
```



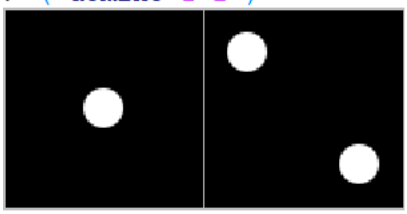
```
> ( domino 0 5 )
```



```
> ( domino 4 3 )
```



```
> ( domino 1 2 )
```



```
>
```

Code:

```
#lang racket

(  require 2htdp/image )

;------------------------------

;

; - Variables to denote the side of a tile and the demensions of a pip

;
```

```
;


( define side-of-tile 100 )
( define diameter-of-pip ( * side-of-tile 0.2 ) )
( define radius-of-pip ( / diameter-of-pip 2 ) )


;------------------------------
;
; - d and nd are used as offsets in the overlay/offset function
applications
;
;


( define d ( * diameter-of-pip 1.4 ) )
( define nd ( * -1 d ) )


;------------------------------
;
; - Bind one variable to a blank tile and another to a pip
;
;


( define blank-tile ( square side-of-tile "solid" "black" ) )
( define ( pip ) ( circle radius-of-pip "solid" "white" ) )


;------------------------------
;
; - Bind one variable to each of the basic tiles
;
;
```

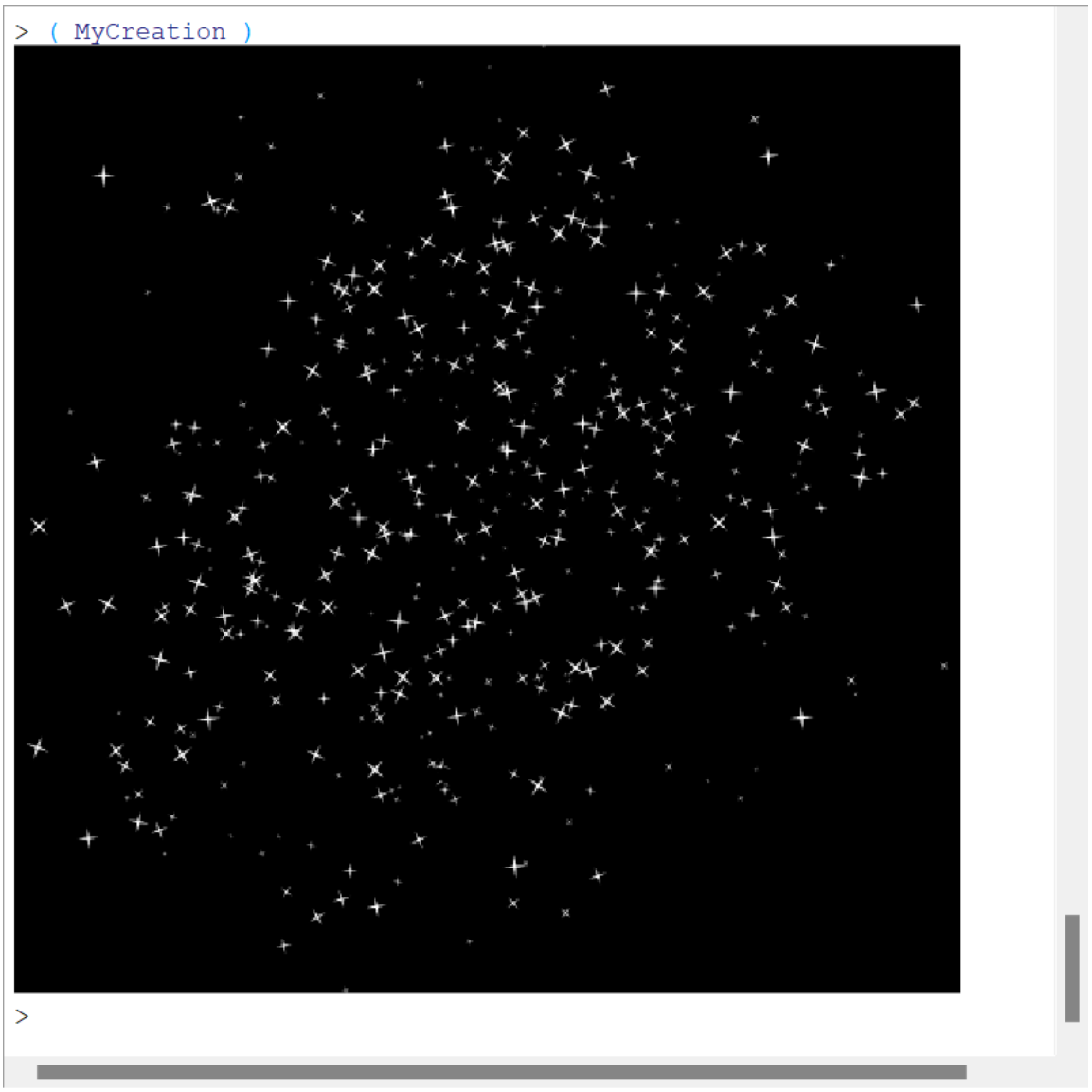
```
( define basic-tile1 ( overlay ( pip ) blank-tile ) )

( define basic-tile2
   ( overlay/offset ( pip ) d d
                    ( overlay/offset ( pip ) nd nd blank-tile )
                    )
   )

( define basic-tile3 ( overlay ( pip ) basic-tile2 ) )

( define basic-tile4a
   ( overlay/offset ( pip ) nd d
                    ( overlay/offset ( pip ) nd nd blank-tile )
                    )
   )
( define basic-tile4b
   ( overlay/offset ( pip ) d nd basic-tile4a )
   )
( define basic-tile4
   ( overlay/offset ( pip ) d d basic-tile4b )
   )

( define basic-tile5
   ( overlay ( pip ) basic-tile4 )
   )
( define basic-tile6a
   ( overlay/offset ( pip ) 0 d basic-tile4 )
   )
( define basic-tile6
   ( overlay/offset ( pip ) 0 nd basic-tile6a )
```

```
   )
;-------------------------------
;
; - Bind one variable to each of the six framed tiles
;
;

( define frame ( square side-of-tile "outline" "gray" ) )

( define tile0 ( overlay frame blank-tile ) )
( define tile1 ( overlay frame basic-tile1 ) )
( define tile2 ( overlay frame basic-tile2 ) )
( define tile3 ( overlay frame basic-tile3 ) )
( define tile4 ( overlay frame basic-tile4 ) )
( define tile5 ( overlay frame basic-tile5 ) )
( define tile6 ( overlay frame basic-tile6 ) )

;-------------------------------
;
; - Function to generate a domino
;
;

( define ( domino a b )
   ( beside ( tile a ) ( tile b ) )
   )
( define ( tile x )
   ( cond
      ( ( = x 0 ) tile0 )
      ( ( = x 1 ) tile1 )
```

```
      ( ( = x 2 ) tile2 )
      ( ( = x 3 ) tile3 )
      ( ( = x 4 ) tile4 )
      ( ( = x 5 ) tile5 )
      ( ( = x 6 ) tile6 )
     )
  )
```

Interaction: Creation

Demo:



Code:

```
#lang racket
( require 2htdp/image )
( define ( MyCreation )
   ( define lower-bound 20 )
   ( define upper-bound 30 )
   ( define num ( rand lower-bound upper-bound ) )
   ( underlay ( background 500 )
   ( column-of-stars num num ) )
   )
( define ( rand low up)
   ( define num ( + ( random up ) low ) )
   num
   )
( define ( row-of-stars n )
   ( cond
      ( ( = n 0 )
        ( display "\n" )
        )
      ( ( = n 1 )
        ( hspace )
        )
      ( ( > n 1 )
        ( rotate 15 ( overlay/offset ( paint-star ) ( rand 0 60 ) ( rand 0 150 ) ( row-of-stars ( - n 1 ) ) ) )
        )
      )
   )
( define ( column-of-stars r c )
   ( cond
```

```
      ( ( = c 0 )
        ( display "\n" )
        )
      ( ( = r 1 )
        ( row-of-stars ( rand 10 35 ) )
        )
      ( ( > r 1 )
        ( rotate 10 ( overlay/offset ( row-of-stars c ) ( rand -150
140 ) ( rand -60 150 ) ( column-of-stars ( - r 1 ) c ) ) )
        )
      )
   )
( define ( vspace )
   ( define lower-bound 25 )
   ( define upper-bound 25 )
   ( define len ( rand lower-bound upper-bound ) )
   ( define the-space ( rectangle len len "solid" "black" ) )
   the-space
   )

( define ( hspace )
   ( define lower-bound 2 )
   ( define upper-bound 10 )
   ( define len ( + ( random upper-bound ) lower-bound ) )
   ( define the-space ( rectangle len len "solid" "black" ) )
   the-space
   )
( define ( paint-star )
   ( define ( star )
      ( define lower-bound 2 )
      ( define upper-bound 8 )
```

```
      ( define len ( + ( random upper-bound ) lower-bound ) )
      ( define the-star ( rotate 45 ( pulled-regular-polygon len 4 1
50 "solid" "white" ) ) )
      the-star
      )
   ( star )
   )
( define ( background num )
   ( define length num )
   ( define ( img ) ( square num "solid" "black" ) )
   ( img )
   )
```