Group: Alen Sugimoto, Francesco Costa, Diell Kryeziu, Laurenz Ebi

Fury Breakout 2020

A variant of Super Breakout in Progressive game mode

Functionality

Layout

When a player starts a new game in the interactive program, the canvas will look something like Fig. 1. The canvas is divided into 32 rows and 13 columns, and the blocks and balls will change color depending on where they are on the canvas: blue in rows 1-4, orange in rows 5-8, green in rows 9-12, yellow in rows 13-16, and white in all the other rows. The 4th-to-last row is reserved for the paddle. The last 3 give some room to display the player's score and the number of lives left.

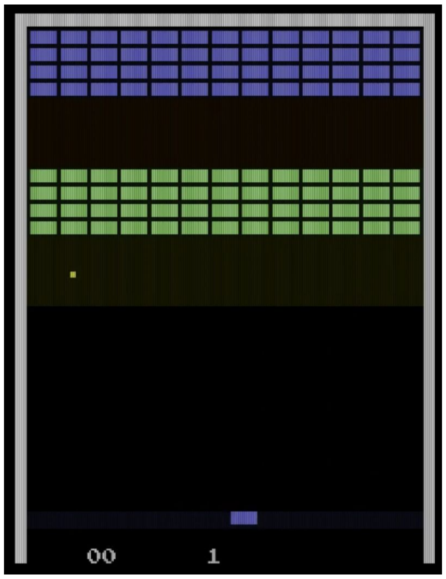

Figure 1. A screenshot taken during the first few seconds of gameplay of Super Breakout in Progressive game mode, from IKINNIK. "IKINNIK Play's SUPER BREAKOUT Progressive Mode." *YouTube*, 20 May 2020, www.youtube.com/watch?v=DpovgdAntJA.

General

The objective of the game is to, before losing all your lives, break as many blocks as possible by using the walls and the paddle to bounce one or more balls into the blocks.

The game is played by a single player. The player has multiple lives and loses one of them when all balls leave the bottom of the canvas.

The blocks progressively move down by 1 row, and new blocks are added to the canvas from above. Four rows of blocks are always separated by four empty rows. Blocks will disappear once they enter the row right above the paddle. Also, at the end of every turn (after losing a life), the first 4 rows above the paddle will be cleared.

Points are scored by breaking blocks. The blocks break when hit by a ball once. The point values of the blocks, which are determined by color, are as follows: 7 for blue, 5 for orange, 3 for green, and 1 for yellow and white.

The paddle will be an ellipse, and thus, the angle the ball rebounds off the paddle depends on which part of the paddle it hits.

Every ball initially has a normal speed and can later independently speed up three times with the following events: 4th paddle hit, 8th paddle hit, and 12th paddle hit.

Regarding game controls, the left and right arrow keys (or the A and D keys) will move the paddle horizontally. The space bar will be the pause/play key, which will open or close a menu once pressed. If a menu is open, the player may be able to use the R key to start a new game, the ESC key to restart the program, or the Q key to stop the program's clock.

We also intend to implement sound that will be triggered by certain events.

Special Features

- When the ball hits the top of the canvas, the paddle's size is reduced to half its original size. It returns to its normal size after every turn.
- A ball's speed is maximized when one of the blocks in the top 8 rows is hit by it.
- The blocks progressively move down by 1 row at an increasing rate, which is reset every turn; in a single turn, the first progression occurs after 8 paddle hits, the next 4 occur after 4 paddle hits each, the next 16 occur after 2 paddle hits each, and the rest occur after 1 paddle hit each.
- When there are more than one ball in play, the point value of every block is multiplied by the number of balls in play.
- When a block breaks, a power-up or a power-down may fall down. The power-ups/downs take effect when caught by the paddle. Possible power-ups include doubling the paddle size, forming a row of blocks below the paddle, and placing an extra ball into play. Possible power-downs include shrinking the paddle size, slowing down the paddle, and increasing the speed of all the balls in play.

Resources

- The rsound library will be used to produce sounds in the game. We decided not to use the racket/gui/base library since it conflicts with the 2htdp/image library.
- WAV files may be used along with sounds made directly with the rsound library.
- The 2htdp/image library will be used to make images.
- The 2htdp/universe library will be used to make our program interactive.

Main Data Types

The following data types are defined to keep track of our program's "world state":

```
; a NonnegativeNumber is a Number greater than or equal to zero
; interpretation: a non-negative number

; a NonnegativeInteger is one of the following:
; - 0                         ; zero
; - (add1 NonnegativeInteger) : a positive integer
; interpretation: a non-negative integer

; a Posn is (make-posn Number Number)
; interpretation: a position vector ('x', 'y') in pixels
; (define-struct posn [x y])
```

```racket
; a Velo is (make-velo Number Number)
; interpretation: a velocity vector ('x', 'y') in pixels per clock tick
(define-struct velo [x y])

; a Ball is (make-ball Posn Velo NonnegativeInteger)
; interpretation: a ball, which has hit the paddle
;                 'paddle-hit-count' times, with
;                 position 's' and velocity 'v'
(define-struct ball [s v paddle-hit-count])

; a Paddle is (make-paddle posnx velox length)
;    where posnx  : Number
;          velox  : Number
;          length : NonnegativeNumber
; interpretation: a paddle with a length of 'length' pixels
;                 and a horizontal position and velocity
;                 of 'posnx' (in pixels)
;                 and 'velox' (in pixels per clock tick)
(define-struct paddle [posnx velox length])

; a Cell is a one of the following:
; - 0 ; an empty cell
; - 1 ; a cell containing a block
; interpretation: a cell that may contain a block

; an Item is (make-item Symbol Posn)
; interpretation: a game item with name 'name' and position 's'
(define-struct item [name s])

; a Game is (make-game lob paddle loloc loi score turnnum delay pause? show?)
;    where lob     : List<Ball>
;          paddle  : Paddle
;          loloc   : List<List<Cell>>
;          loi     : List<Item>
;          score   : NonnegativeInteger
;          turnnum : NonnegativeInteger
;          delay   : NonnegativeInteger
;          pause?  : Boolean
;          show?   : Boolean
; interpretation: a game with list of Balls 'lob',
;                 Paddle 'paddle', table of Cells 'loloc',
;                 list of Items 'loi', a score of 'score' points,
;                 a turn number of 'turnnum', a delay of 'delay' seconds, and
;                 'pause?' and 'show?' to indicate whether or not
;                 it should pause and show itself
(define-struct game [lob paddle loloc loi score turnnum delay pause? show?])

; a World is (make-world game stop?)
;    where game  : Game
;          stop? : Boolean
; interpretation: a world with Game 'game' and 'stop?' to indicate whether
;                 or not it should stop the clock
(define-struct world [game stop?])
```