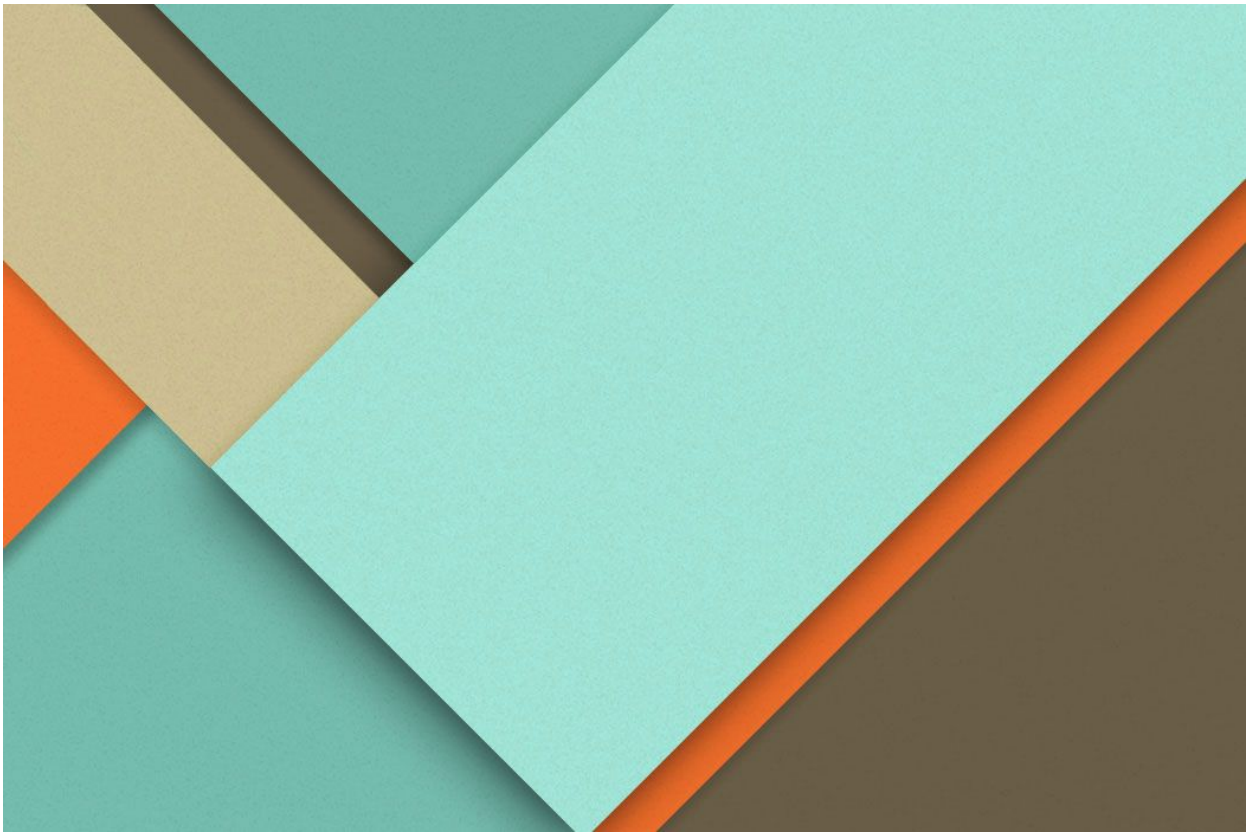


PROJECT REPORT

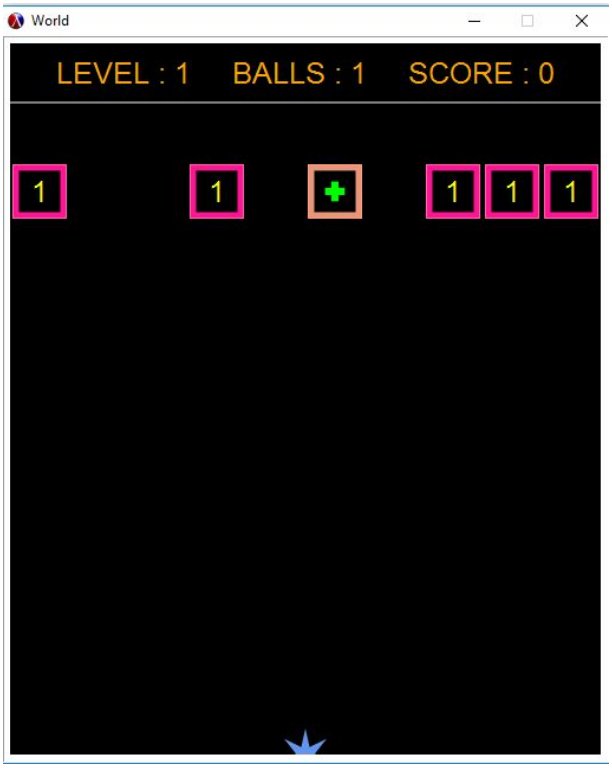
BRICKBREAKER

AMAN JAIN (160050019)

SHUBHAM ANAND (160050060)

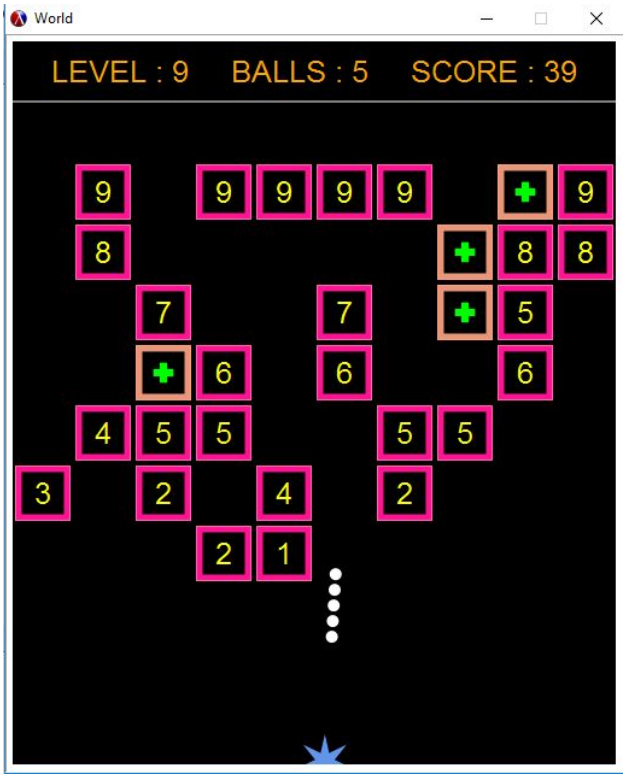
NEELESH VERMA (160050062)

Description of the problem :-



BRICK-BREAKER is the game where you need to aim the target in order to throw the balls or to break the bricks. The angle is a key point and this is a simple but addictive game.

To play, BRICK-BREAKER game you have to control the gun. You will be throwing the balls from a bottom of a screen upwards. The mission is to break the blocks which appear from an upper side of a screen by hitting them with the balls that the gun is going to throw. To fire the balls, you have to click at any point of a screen and the balls are fired in that direction. There are blocks on screen having numbers on them that indicate the number of times that they have to be hit up so that they can disappear completely.



For example,if a square has number three on it, so when you shoot four balls on it, first three will hit the square and they will bounce back at the angle. The last one is going to go through the dissolved square and it can hit other shapes that will be found in its way. It will then be bouncing back on its way until it can end up by getting at the bottom of a screen once again. It is not important catching the ball when it has bounced back. Balls can just hit a bottom and then they may appear afterwards in the gun.There is a ball to throw at every turn and balls are added by hitting the plus symbol while floating around a screen.

The game ends whenever the blocks (at-least one) touches the bottom.

Design of the program -

1. The package **2htdp/image** and **2htdp/universe** have been used for animation.
2. Two structures have been used in the program - ball and block, ball has position and velocity and the block has index and value.The index of block gives its position in list.Plus is represented as a block with value -1.
   The playing area is divided into grids and the blocks are represented by a list of struct blocks.The square of grid having no block is represented by struct block with value 0.
3. The state of the game is depicted by :-
   consed-pair , which is of the form - (cons list-of-balls list-of-objects(blocks)),
   Level, Score, Number-of-balls, gun-position.
4. The program can broadly be divided in two parts :-
   One part, **move-balls** takes consed-pair as argument, it then returns the state of the game at the next instant taking into account the collisions of balls with the blocks.
   Another part, **big-bang** simulates the animation by drawing the world after every clock-tick.
5. Move-balls has the following functions :-
   Collide? :- takes a ball as argument and checks at that instant whether it is colliding with any block or not and gives the result(#f , 'top, 'bottom, 'right, 'left) to the next function move-ball.

Move-ball :- takes a ball as argument and using the result of collide? Changes the velocity if needed.
Next-posn :- takes ball as argument and changes the position vector of ball.

6. **Check-bottom** :- takes list of balls as argument, checks whether it touches the bottom and removes the ones which touch bottom.It also sets the gun-position as the position of the first ball to hit the bottom.
7. **Big-bang** :- takes as argument -
   Initial-state
   And the following four functions -
   - on-tick : takes move-balls and tick-time as arguments and applies move-balls to consed-pair after every tick-time.
   - to-draw : takes place-everything as argument, which draws the world after each tick.
   - on-mouse : takes mouse-function as argument, which takes combined-pair, coordinates of mouse click along with an event which gives the direction for the shoot.
   - stop-when : takes a function game-over? And a world as argument, whenever the **game-over?** Evaluates to #t the game is stopped(game-over) and produces the world (game-over-screen).

Sample input-output :-

- The game is started by a command **(play-game)**.
- **Mouse-click** : At beginning of each level, user clicks on the screen and its coordinates are used by the gun to aim and shoot balls.

Limitations and Bugs :-

- The game becomes slow at very high levels (number of balls become too large).
- We could not represent the projector lines (line of aim/shoot).
- In very rare situations, a ball passes through a block.

Use of Abstractions and Higher order functions :-

- **Place-images** : takes list of images along with their coordinates and a background as arguments and places all the images on the background.
- A helper function used inside collide? is an abstraction which takes the index of block which might be colliding with the ball and the result it has to return in case of collision.
- On-tick, to-draw, on-mouse, stop-when are all examples of higher order functions.
- Abstractions on list such as Foldr and map have been used extensively wherever found useful.