STACKER Gaming Application

END-SEM PROJECT REPORT

Submitted by

Group -E3

B.SAI ABHISHEK	**-**	**BL.EN.U4AIE21015**
RUCHITH BALAJI	**-**	**BL.EN.U4AIE21017**
CHILAKURU HARI	**-**	**BL.EN.U4AIE21038**

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

Elements of Computing Systems-2

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE – 560035

July -2022.

TABLE OF CONTENTS

ABSTRACT

- The STACKER game is totally written JACK language.
- To run the game in VM Emulator.
- A Player can stop the Moving Row Horizontally by pressing the SPACEBAR and place it on the Top of the Layer.
- When the Player places 15 layers continuously…Then the player Wins!! And the game restarts.
- Stacker is an arcade game where the goal is to build a stack of blocks as high as possible.
- Stacker game contains 15 Levels.
- At each level, a row of blocks moves sideways and the user has to lock the blocks in place. And timing it so that it aligns with the previous level.
- Blocks that don't align are lost and if no blocks aligned at all, the player loses.
  As the levels increase, the blocks move faster making timing even more critical.

DETAILED DESCRIPTION

Jack Programming Language:

- Jack is an object-based language like Java.
- It is a multipurpose language and a high-level language.
- Can be used to write complex programs like Operating System.
- Uses methods, constructors and functions.
- Can create complex data structures like lists, trees.
- Screen is divided into 256 rows and 512 columns.
- The screen left top most corner starts from (0,0), right end corner (511,255).
- Screen.drawRectangle(x1,x2,y1,y2)- (x1,x2)&(y1,y2) are coordinates of diagonals of rectangle.
- Screen.drawLine(x1,x2,y1,y2)- (x1,x2)&(y1,y2) are coordinates of ends of lines.
- Sys.wait(x) – For delaying for few seconds
- Screen.setColor(false) –sets screen to white
- Screen.setColor(true) –sets screen to black

Built-in Functions:

- Output.printString()- To print a string
- Keyboard.keypressed()-returns the key pressed
- Screen.drawPixel(x,y) -x:row,y:column
- Memory.poke(x,y) – does 16 memory write operations starting from x bit in y column
- Memory.deAlloc(this)- to deallocate the memory and recycle
- Keyboard.readLine()/Keyboard.readInt() to take input from user

Used Jack Files:

In this game we have used 7 jack files to Run:

- Stacker.jack
- Constants.jack
- Drawer.jack
- Stack.jack
- MovingRow.jack
- Stackergame.jack
- Main.jack

Stacker.jack

- This jack file creates a Start page for player to proceed.
- In this File Constructer Stacker is used.
- An array of length is initialised to print the Commands or Menu for player to proceed further.
- Two keys are used with appropriate method to start and Quit the game.
- Takes the User Input from player and corresponding work with assigned key will be done

Constants.jack

- In this file 5 functions to perform the play and quit Operations on the Start Menu.
- The functions are COLS , LEVELS , KEY_SPACE , KEY_P , KEY_Q.
- KEY_P = Starts the game ; KEY_Q = Quits the game

Drawer.jack

- This file Draws a grid with Built-in methods where the stacker game is to be performed.
- This file also draws the Row which is to played during the Game.

Stack.jack

- In this file a Constructer is created With named as Stack.
- And Initialised the array for Creating the Required levels.
- Adds a new row to the stack, keeping only blocks that are stackable.

MovingRow.jack

- Implements the row that moves sideways.
- In this File set the number of blocks (aBlocks) starting from index (offset) in the row.
- Sets the moving speed and block starting position according to the new level
- Set the speed of the blocks given a level.
- Determines whether to move the blocks and where to move them

Main.jack

- Creates a new Stacker game.
- Stacker is disposed.

Stackergame.jack

- Implements a stacker game.
- Stacker is an arcade game where the goal is to build a stack of blocks as high as possible. At each level, a row of blocks moves sideways.
- The user has to lock the blocks in place (using the SPACE key) and timing it so that it aligns with the previous level.
- Blocks that don’t align are lost and if no blocks aligned at all, the player loses.
- As the levels increase, the blocks move faster making timing even more critical.
- Acts as the controller between moving the row, updating the stack, the game state and drawing to screen

CODE:
Stacker.jack

```
class Stacker {
  field Array menuStr;
  field boolean quit;
  field int key;
  field StackerGame game;

  constructor Stacker new() {
    let menuStr = Array.new(6);
    let menuStr[0] = "S T A C K E R";
    let menuStr[1] = "-------------";
    let menuStr[2] = "Stack blocks by pressing SPACE to lock the row in place";
    let menuStr[3] = "Press 'P' to play, 'Q' to quit.";
    let menuStr[4] = "Created By Abhishek,Ruchith,Hari.";
    let menuStr[5] = "Group Name = E3";
    return this;
  }
  method void run() {
    while (~(key = Constants.KEY_Q())) {
      do Screen.clearScreen();
      do Output.moveCursor(10, 26);
      do Output.printString(menuStr[0]);
      do Output.moveCursor(11, 26);
      do Output.printString(menuStr[1]);
      do Output.moveCursor(13, 4);
      do Output.printString(menuStr[2]);
      do Output.moveCursor(15, 17);
      do Output.printString(menuStr[3]);
      do Output.moveCursor(22, 17);
      do Output.printString(menuStr[4]);
      do Output.moveCursor(2, 1);
      do Output.printString(menuStr[5]);

      while (key = 0) {
        let key = Keyboard.keyPressed();
      }

      if (key = Constants.KEY_P()) {
        do Screen.clearScreen();
        let game = StackerGame.new();
        do game.run();
        do game.dispose();
        let key = 0;
      }
    }
    return;
  }
```

Constants.jack

```
class Constants {
  /** number of columns in the game */
  function int COLS() {
    return 7;
  }
  /** number of levels in a game */
  function int LEVELS() {
    return 15;
  }
  /** ascii code for space key */
  function int KEY_SPACE() {
    return 32;
  }
  /** ascii code for P key */
  function int KEY_P() {
    return 80;
  }
  /** ascii code for Q key */
  function int KEY_Q() {
    return 81;
  }
}
```

Main.jack

```
class Main {
  function void main() {
    var Stacker stacker;
    let stacker = Stacker.new();
    do stacker.run();
    do stacker.dispose();
    return;
  }
}
```

Drawer.jack

```
class Drawer {

  function void grid() {
    var int x, y, i, j;
    let x = 208;
    let y = 226;
    do Screen.setColor(true);

    while (i < 8) {
      do Screen.drawLine(x, 16, x, 226);
      let x = x + 14;
      let i = i + 1;
    }

    while (j < 16) {
      do Screen.drawLine(208, y, 306, y);
      let y = y - 14;
      let j = j + 1;
    }
    return;
  }

  function void row(Array row, int level) {
    var int col;
    let col = 0;

    while (col < Constants.COLS()) {
      if (row[col]) {
        do Drawer.block(col, level, true);
      } else {
        do Drawer.block(col, level, false);
      }
      let col = col + 1;
    }
    return;
  }

  function void block(int xoff, int yoff, boolean isBlack) {
    var int i, addr, x, y;
    let i = 0;
    let addr = 16896;
    let x = 210 + (xoff * 14);
    let y = 214 - (yoff * 14);

    do Screen.setColor(isBlack);
    do Screen.drawRectangle(x, y, x + 10, y + 10);
    return;
  }
}
```

Stack.jack

```
class Stack {
  field Array stack;
  constructor Stack new() {
    var int r;
    let stack = Array.new(Constants.LEVELS());
    let r = 0;
    while (r < Constants.LEVELS()) {
      let stack[r] = Array.new(Constants.COLS());
      let r = r + 1;
    }
    return this;
  }
  method int add(Array row, int level) {
    var int blocksStacked, i;
    var Array top, newTop;
    let top = stack[level - 1];
    let newTop = stack[level];
    let i = 0;
    let blocksStacked = 0;

    if (level = 0) {
      let newTop = stack[0];
      while (i < Constants.COLS()) {
        let newTop[i] = row[i];
        let i = i + 1;
      }
      let blocksStacked = 3;
    } else {
      while (i < Constants.COLS()) {
        let newTop[i] = top[i] & row[i];

        if (newTop[i]) {
          let blocksStacked = blocksStacked + 1;
        }
        let i = i + 1;
      }
    }
    return blocksStacked;
  }

  method Array getRow(int level) {
    return stack[level];
  }
  method void dispose() {
    do Memory.deAlloc(this);
    return;
  }
}
```

MovingRow.jack

```
class MovingRow {
  field int x; // starting index for blocks
  field int delay; // block moving speed. smaller value is faster
  field int direction; // 1 = move right, -1 = move left
  field int time; // counter used to determine when to move the blocks
  field int blocks; // number of blocks in the row
  field Array row;
  constructor MovingRow new() {
    let delay = 1000;
    let time = 0;
    let direction = 1;
    let row = Array.new(Constants.COLS());
    do setRow(2, 3);
    do setLevel(0, blocks);
    return this;
  }
  method void setRow(int offset, int aBlocks) {
    var int i;
    let i = 0;
    let x = offset;
    let blocks = aBlocks;

    while (i < Constants.COLS()) {
      if ((i > (x - 1)) & (i < (x + blocks))) {
        let row[i] = true;
      } else {
        let row[i] = false;
      }
      let i = i + 1;
    }
    return;
  }

  method Array getRow() {
    return row;
  }
  method void setLevel(int level, int aBlocks) {
    do setDelay(level);
    do setRow(2, aBlocks);
    return;
  }

  /** set the speed of the blocks given a level */
  method void setDelay(int level) {
    if (level = 1) {
      let delay = 200;
    }

    if ((level > 1) & (level < 10)) {
      let delay = delay - 10;
    }
    if (level = 10) {
      let delay = 95;
    }

    if (level > 10) {
      let delay = delay - 30;
    }

    return;
  }

  method void move() {
    if (time < delay) {
      let time = time + 1;
      return;
    } else {
      let time = 0;
    }

    if (x = (Constants.COLS() - blocks)) {
      let direction = -1;
    }

    if (x = 0) {
      let direction = 1;
    }

    let x = x + direction;
    do setRow(x, blocks);
    return;
  }

  method void dispose() {
    do Memory.deAlloc(this);
    return;
  }
}
```

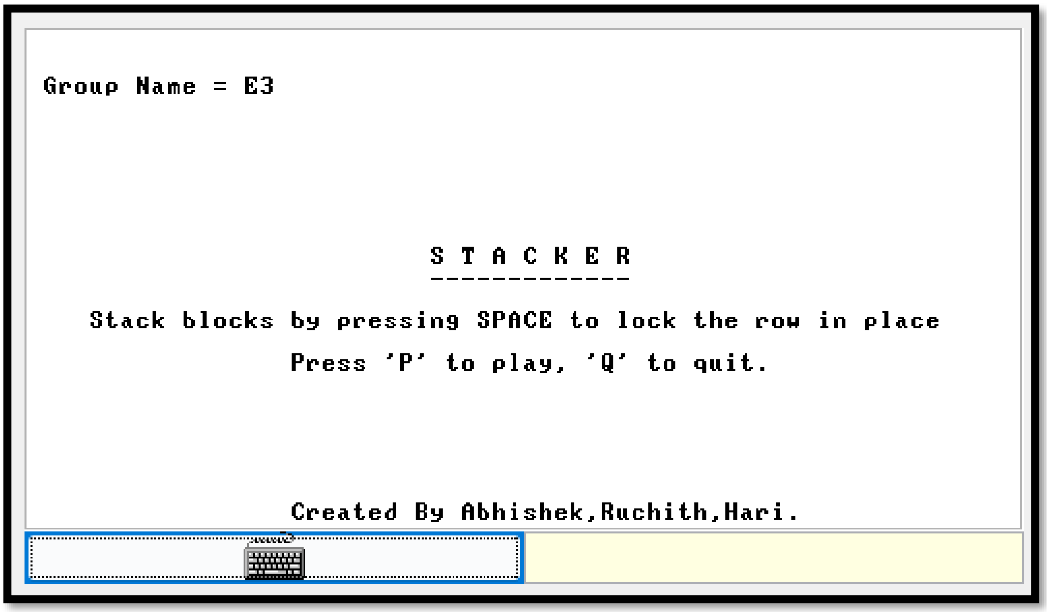
Stackergame.jack

```
class StackerGame {
  field int level;
  field MovingRow mover;
  field Stack stack;
  field boolean play;

  constructor StackerGame new() {
    let level = 0;
    let mover = MovingRow.new();
    let stack = Stack.new();
    let play = true;
    return this;
  }

  method void run() {
    var char key;
    var int blocks;
    var String levelStr;
    let levelStr = String.new(2);
    do Drawer.grid();
    do Output.moveCursor(21, 26);
    do Output.printString("S T A C K E R");

    while (play) {
      do levelStr.setInt(level + 1);
      do Output.moveCursor(1, 1);
      do Output.printString("Level ");
      do Output.printString(levelStr);
      do Output.printString("/");
      do Output.printInt(Constants.LEVELS());

      while (~(key = Constants.KEY_SPACE())) {
        let key = Keyboard.keyPressed();
        do mover.move();
        do Drawer.row(mover.getRow(), level);
      }

      while (key = Constants.KEY_SPACE()) {
        let blocks = stack.add(mover.getRow(), level);
        do Drawer.row(stack.getRow(level), level);

        do gameState(blocks);
        let key = 0;
        do Sys.wait(1000);
      }
    }
    return;
  }
  method void gameState(int blocks) {
    if (blocks = 0) {
      let play = false;
      do Output.moveCursor(3, 1);
      do Output.printString("You lost the Game");
      do Output.moveCursor(4, 1);
      do Output.printString("Please Try Again");
      do Sys.wait(2000);
      do Output.moveCursor(6, 1);
      do Output.printString("Returning to Interface");
      do Sys.wait(2000);
    } else {
      if (level = 14) {
        let play = false;
        do Output.moveCursor(3, 1);
        do Output.printString("You win!");
        do Output.moveCursor(4, 1);
        do Output.printString("Congratulations");
        do Sys.wait(2000);
        do Output.moveCursor(6, 1);
        do Output.printString("Returning to Interface");
        do Sys.wait(2000);
      } else {
        let play = true;
        let level = level + 1;
        do mover.setLevel(level, blocks);
      }
    }
    return;
  }

  method void dispose() {
    do mover.dispose();
    do stack.dispose();
    do Memory.deAlloc(this);
    return;
  }
}
```

OUTPUT:

INTERFACE:

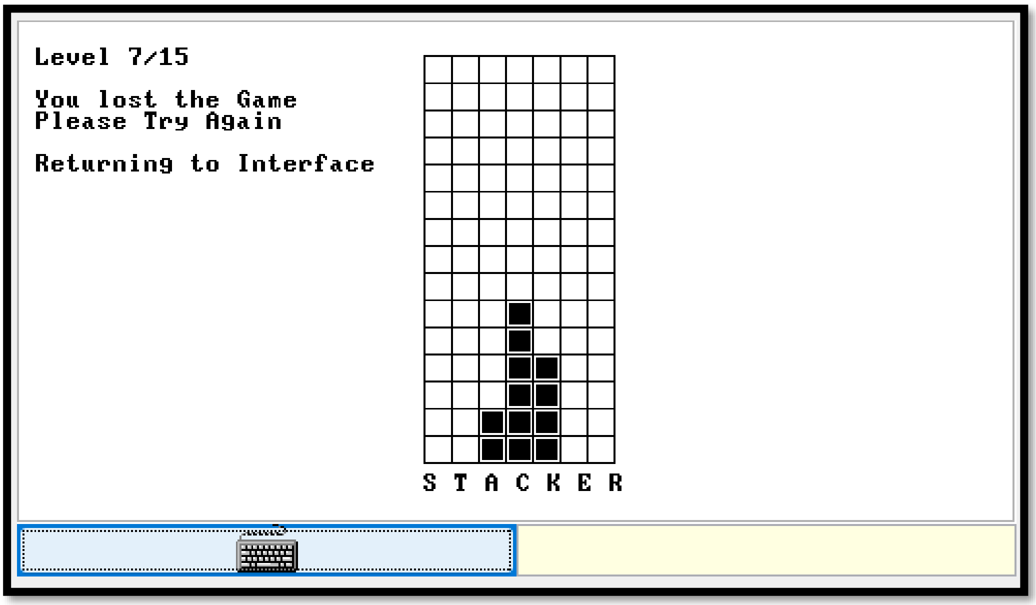
At Start:



```
Group Name = E3

                    S T A C K E R
                    -------------
   Stack blocks by pressing SPACE to lock the row in place
              Press 'P' to play, 'Q' to quit.



              Created By Abhishek,Ruchith,Hari.
```

At End:

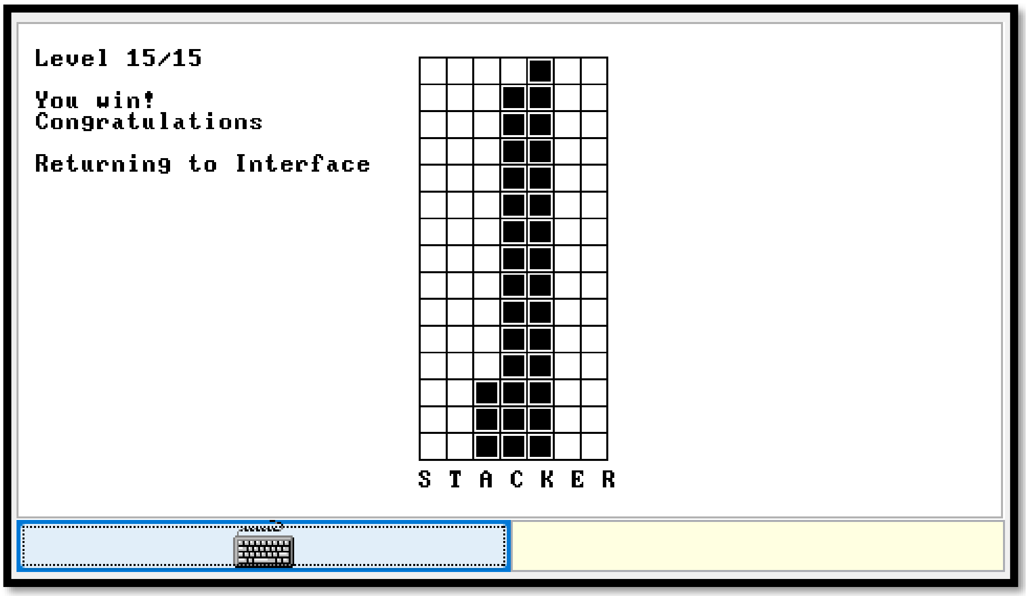
- ✓ Case-1: (If player Loses)



```
Level 7/15

You lost the Game
Please Try Again

Returning to Interface
```

After Winning the game and the game will restarts.

✓ Case-2: (If player Wins)



```
Level 15/15

You win!
Congratulations

Returning to Interface
```

After Losing the game and the game will restarts.
The **SCORE** is displayed at the Top-left corner of the Screen.

THANK YOU