THE PING PONG GAME

Report For the submission towards End-Sem Project

19AIE112 - Elements of Computing Systems 2

Team Members :

1.) BL.EN.U4AIE19014 - **CHAVALI SURYATEJA**

2.) BL.EN.U4AIE19041 - **MATURI TANUJ**

3.) BL.EN.U4AIE19053 - **RAMA SAI ABHISHEK P**

4.) BL.EN.U4AIE19065 - **T. SAI ESWAR REDDY**

Submitted by: **CHAVALI SURYA TEJA** (BL.EN.U4AIE19014)

ABSTRACT

Description :

This is a two-player Ping pong game. The main theme of the game is to strike the ball with the bat, while the bat moving only in the vertical direction. Initially, the start screen displays two options:

1.) Start the Game

2.) Controls for the game

By selecting the start game option (1), we directly jump into the game, whereas, when the any of the players are not sure of the controls related to the game, they can choose to see the controls page (2), which briefly explains the keys used to operate the bats, etc.

Since this is a two-player game, the two main inputs would be from the 2 players, while moving the two bats only in the vertical direction, trying to hit the ball from opposite directions respectively. The output would be the score counted on the screen. The first player to score 5 points wins the game!

Brief description of all the classes in the project.

Class Sprite:

In this class, we will generate 2 Smiley sprites using the Bit Map Editor.

Subroutines :

- Dispose()
- Smiley1()
- Smiley2()

Code:

```
class Sprite {

 field int x, y;

  constructor Sprite new(int a , int b){
   let x =a;
   let y=b;
   return this;
}

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

method void Smiley1() {
```

```
        var int memAddress;
    let memAddress = 16384+ (x/16)+ (y*32);
    do Memory.poke(memAddress+0, 960);
    do Memory.poke(memAddress+32, 1056);
    do Memory.poke(memAddress+64, 2064);
    do Memory.poke(memAddress+96, 4104);
    do Memory.poke(memAddress+128, 8772);
    do Memory.poke(memAddress+160, 16386);
    do Memory.poke(memAddress+192, -32767);
    do Memory.poke(memAddress+224, -28663);
    do Memory.poke(memAddress+256, -28663);
    do Memory.poke(memAddress+288, -30703);
    do Memory.poke(memAddress+320, 17442);
    do Memory.poke(memAddress+352, 9156);
    do Memory.poke(memAddress+384, 4104);
    do Memory.poke(memAddress+416, 2064);
    do Memory.poke(memAddress+448, 1056);
    do Memory.poke(memAddress+480, 960);

        return;
}

method void Smiley2() {
        var int memAddress;
    let memAddress = 16384+(x/16)+ (y*32);
```

```
        do Memory.poke(memAddress+0, 960);
        do Memory.poke(memAddress+32, 1056);
        do Memory.poke(memAddress+64, 2064);
        do Memory.poke(memAddress+96, 4104);
        do Memory.poke(memAddress+128, 8772);
        do Memory.poke(memAddress+160, 16386);
        do Memory.poke(memAddress+192, -32767);
        do Memory.poke(memAddress+224, -28663);
        do Memory.poke(memAddress+256, -28663);
        do Memory.poke(memAddress+288, -30703);
        do Memory.poke(memAddress+320, 17442);
        do Memory.poke(memAddress+352, 9156);
        do Memory.poke(memAddress+384, 4104);
        do Memory.poke(memAddress+416, 2064);
        do Memory.poke(memAddress+448, 1056);
        do Memory.poke(memAddress+480, 960);
        return;
}
}
```

Class Up:

In this class, we will generate both the DownArrow and the UpArrow sprites using the Bit Map Editor.

Subroutines :

- Dispose()
- UpArrow()
- DownArrow()

Code:

```
class Up {

 field int x, y;

  constructor Up new(int a , int b){
   let x =a;
   let y=b;
   return this;
}

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

method void UpArrow() {
                var int memAddress;
          let memAddress = 16384+(x/16) + (y*32);
          do Memory.poke(memAddress+0, 128);
          do Memory.poke(memAddress+32, 448);
          do Memory.poke(memAddress+64, 672);
          do Memory.poke(memAddress+96, 1168);
          do Memory.poke(memAddress+128, 2184);
```

```
        do Memory.poke(memAddress+160, 4228);
        do Memory.poke(memAddress+192, 8322);
        do Memory.poke(memAddress+224, 128);
        do Memory.poke(memAddress+256, 128);
        do Memory.poke(memAddress+288, 128);
        do Memory.poke(memAddress+320, 128);
        do Memory.poke(memAddress+352, 128);
        do Memory.poke(memAddress+384, 128);
        do Memory.poke(memAddress+416, 128);
        do Memory.poke(memAddress+448, 128);
        do Memory.poke(memAddress+480, 128);
        return;
}

method void DownArrow() {
            var int memAddress;
        let memAddress = 16384+(x/16) + (y*32);
        do Memory.poke(memAddress+0, 128);
        do Memory.poke(memAddress+32, 128);
        do Memory.poke(memAddress+64, 128);
        do Memory.poke(memAddress+96, 128);
        do Memory.poke(memAddress+128, 128);
        do Memory.poke(memAddress+160, 128);
        do Memory.poke(memAddress+192, 128);
        do Memory.poke(memAddress+224, 128);
```

```
        do Memory.poke(memAddress+256, 16513);
        do Memory.poke(memAddress+288, 8322);
        do Memory.poke(memAddress+320, 4228);
        do Memory.poke(memAddress+352, 2184);
        do Memory.poke(memAddress+384, 1168);
        do Memory.poke(memAddress+416, 672);
        do Memory.poke(memAddress+448, 448);
        do Memory.poke(memAddress+480, 128);
        return;
    }
}
```

Class Ball :

This is the class related to one of the main figures in the game which is the ball. The ball class mainly consists of the movement of the ball and the bouncing.
The ball moves randomly in any direction.

Subroutines :

- setX()
- setY()
- dispose()
- show()
- hide()
- getLeft()
- getRight()
- getTop()
- getBottom()
- setDestination()

- move()
- bounce()

Code :

```
class Ball{
    field int x,y;
        field int lengthx, lengthy;
    field int d, straightD, diagonalD;
    field boolean invert, positivex, positivey;
    field int leftWall, rightWall, topWall, bottomWall;
    field int wall;
        constructor Ball new(int Ax, int Ay,
                int AleftWall, int ArightWall, int AtopWall, int AbottomWall) {
          let x = Ax;
          let y = Ay;
          let leftWall = AleftWall;
          let rightWall = ArightWall -6 ;
          let topWall = AtopWall;
          let bottomWall = AbottomWall -6;
          let wall = 0;
      do show();
      return this;
   }
        method void setX(int h)
        {
        let x=h;
                        return;
        }
        method void setY(int u)
```

```
{
let y=u;
                    return;
}


method void dispose(){
do Memory.deAlloc(this);
return;
}
method void show(){
do Screen.setColor(true);
do Screen.drawRectangle(x,y,x+5,y+5);
return;
}
method void hide(){
do Screen.setColor(false);
do Screen.drawRectangle(x,y,x+5,y+5);
return;
}
method int getLeft(){
return x;
}
method int getRight(){
return x+5;
}
method int getTop(){
return y;
```

```
    }
    method int getBottom(){
    return y+5;
    }

    method void setDestination(int destx, int desty) {
var int dx, dy, temp;
        let lengthx = destx - x;
        let lengthy = desty - y;
let dx = Math.abs(lengthx);
let dy = Math.abs(lengthy);
let invert = (dx < dy);

if (invert) {
    let temp = dx; // swap dx, dy
    let dx = dy;
    let dy = temp;
            let positivex = (y < desty);
    let positivey = (x < destx);
}
else {
            let positivex = (x < destx);
    let positivey = (y < desty);
}

let d = (2 * dy) - dx;
let straightD = 2 * dy;
let diagonalD = 2 * (dy - dx);
```

```
            return;
}
method int move() {

            do hide();

    if (d < 0) { let d = d + straightD; }
    else {
        let d = d + diagonalD;

        if (positivey) {
            if (invert) { let x = x + 4; }
            else { let y = y + 4; }
        }
        else {
            if (invert) { let x = x - 4; }
            else { let y = y - 4; }
        }
            }

    if (positivex) {
        if (invert) { let y = y + 4; }
        else { let x = x + 4; }
            }
            else {
        if (invert) { let y = y - 4; }
        else { let x = x - 4; }
```

```
            }

            if (~(x > leftWall)) {
                let wall = 1;
                let x = leftWall;
            }
        if (~(x < rightWall)) {
                let wall = 2;
                let x = rightWall;
            }
        if (~(y > topWall)) {
            let wall = 3;
                let y = topWall;
        }
        if (~(y < bottomWall)) {
            let wall = 4;
                let y = bottomWall;
        }

            do show();

            return wall;
    }

    method void bounce(int bouncingDirection) {
        var int newx, newy, divLengthx, divLengthy, factor;
```

```
        let divLengthx = lengthx / 10;
        let divLengthy = lengthy / 10;
            if (bouncingDirection = 0) { let factor = 10; }
            else {
               if (((~(lengthx < 0)) & (bouncingDirection = 1)) | ((lengthx < 0) &
(bouncingDirection = (-1)))) {
            let factor = 20;
          }
               else { let factor = 5; }
            }

            if (wall = 1) {
               let newx = 506;
               let newy = (divLengthy * (-50)) / divLengthx;
          let newy = y + (newy * factor);
            }
        else {
          if (wall = 2) {
            let newx = 0;
            let newy = (divLengthy * 50) / divLengthx;
            let newy = y + (newy * factor);
               }
               else {
            if (wall = 3) {
                        let newy = 250;
                        let newx = (divLengthx * (-25)) / divLengthy;
               let newx = x + (newx * factor);
                     }
            else {
```

```
                    let newy = 0;
                    let newx = (divLengthx * 25) / divLengthy;
            let newx = x + (newx * factor);
                }
        }
    }

    do setDestination((newx), (newy));
    return;
  }
}
```

Class Pad :

This class is related to 2 pads that have been used in the game.

The Pad class includes the complete logic for the movement of the pad.

Subroutines :

- dispose()
- show()
- hide()
- setDirection()
- getLeft()
- getTop()
- getBottom()
- setWidth()
- getWidth()
- getRight()
- move

Code:

```
class Pad {
            field int x,y;
     field int width, height;
     field int direction;
  constructor Pad new(int Ax,int  Ay,int  w,int  h){
    let  x=Ax;
     let  y=Ay;
     let width=w;
     let height=h;
     let direction=2;
     do show();
      return this;
}

method void dispose(){
    do Memory.deAlloc(this);
       return;
}
method void show(){
     do Screen.setColor(true);
     do Screen.drawRectangle(x,y,x+width,y+height);
     return;
}

method void hide(){
    do Screen.setColor(false);
     do Screen.drawRectangle(x,y,x+width,y+height);
```

```
    return;
}
 method void setDirection(int Adirection) {
       let direction = Adirection;
       return;
}
method int getLeft() {
       return x;
   }

          method int getTop() {
       return y;
   }
          method int getBottom() {
       return (y+height);
   }
          method void setWidth(int l){
          do hide();
          let height=l;
          do show();
          return;
          }
          method int getWidth()
          {   return width;}

   method int getRight() {
       return (x+width);
   }
```

```
 method void move(){
     if(direction=2){
       let  y = y + 4;
         if((y+height)>255){ let y=255-height;}
       do Screen.setColor(false);
       do Screen.drawRectangle(x,y - 4,(x + width),(y+height) - 4 );
       do Screen.setColor(true);
       do Screen.drawRectangle(x,y,x+width,y+height);
}
else{
     let  y = y - 4;
       if(y<42){ let y=43;}
    do Screen.setColor(false);
    do Screen.drawRectangle(x,y + 4,(x + width),(y+height) + 4 );
    do Screen.setColor(true);
    do Screen.drawRectangle(x,y,x+width,y+height);
   }
    return;
}
}
```

Class Start :

The Start class is the main class related to the Start screen of our Game. It contains the title of the game – " The Ping Pong Game", two Smiley sprites on either side and :

1.) Start Game – Press Space

2.) Controls - Press X

Subroutines :

- dispose()

Code:

```
class Start {

    field Sprite t;

    field Sprite s;

    field char key1, key2;

constructor Start new() {

    let key1 = 62;

    let key2 = 45;

    let t =Sprite.new(145, 87);

    do t.Smiley1();

    let s =Sprite.new( 360, 87);

    do s.Smiley2();

    do Screen.setColor(true);

    do Screen.drawRectangle( 0 , 0 ,4, 255);
```

```
do Screen.setColor(true);

do Screen.drawRectangle(  0, 0 , 511, 4 );

do Screen.setColor(true);

do Screen.drawRectangle( 507 , 0 , 511, 255);

do Screen.setColor(true);

do Screen.drawRectangle(  0, 251 , 511, 255 );

do Output.moveCursor(16, 16);

do Output.printChar(key2);

do Output.moveCursor(16 , 17);

do Output.printChar(key1);

do Output.moveCursor(12 , 16);

do Output.printChar(key2);

do Output.moveCursor(12 , 17);

do Output.printChar(key1);
```

```
    do Output.moveCursor(16 , 20);

    do Output.printString("Controls    -  Press   X ");

    do Output.moveCursor(12 , 19) ;

    do Output.printString(" Start Game  -  Press Space");

    do Output.moveCursor(8 , 21) ;

    do Output.printString("  The Ping Pong Game  " );

    do Screen.setColor( true);

    do Screen.drawRectangle( 175 , 104 , 332 , 105);

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

}
```

Class Controls :

The Controls class is the main class-related controls screen, which contains all the relevant information about the controls of the left player and right player. It also has an option to go back to the Start screen and start the game.

Subroutines :

- dispose()

Code:

```
class Controls {

  field Up t;

  field Up s;
   constructor Controls new() {

let t = Up.new(336,94);

  do t.UpArrow();

   let s = Up.new(336 ,151);

  do s.DownArrow();

  do Screen.setColor(true);

  do Screen.drawRectangle( 253, 0 , 256 , 195 );

  do Output.moveCursor( 3, 5 );
```

```
do Output.printString(" Left Player Controls  " );

do Screen.setColor( true);

do Screen.drawRectangle( 45 , 52 , 210 , 53);

do Output.moveCursor( 3, 36);

do Output.printString(" Right Player Controls " );

do Screen.setColor( true);

do Screen.drawRectangle( 292 , 52 , 469 , 53);

do Output.moveCursor(9 , 3);

do Output.printString("PRESS   W   TO  MOVE    UP");

do Output.moveCursor(14 , 3);

do Output.printString("PRESS   S   TO  MOVE   DOWN");

do Output.moveCursor(9 , 36);

do Output.printString("PRESS ");

do Output.moveCursor(9 , 45);
```

```
do Output.printString(" TO  MOVE   UP");

do Output.moveCursor(14 , 36);

do Output.printString("PRESS");

do Output.moveCursor(14, 45);

do Output.printString(" TO  MOVE  DOWN");

do Output.moveCursor( 19 , 18);

do Output.printString("  BackSpace -  Game Over....  ");

do Output.moveCursor( 21 , 14);

do Output.printString(" ( Backspace - Go back to Main Menu )");

do Screen.setColor(true);

do Screen.drawRectangle( 0 , 0 ,4, 255);

do Screen.setColor(true);

do Screen.drawRectangle(  0, 0 , 511, 4 );

do Screen.setColor(true);
```

```
  do Screen.drawRectangle( 507 , 0 , 511, 255);

  do Screen.setColor(true);

  do Screen.drawRectangle(  0, 251 , 511, 255 );

  do Screen.setColor(true);

  do Screen.drawRectangle( 0 , 195 , 511 , 198);

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

Class Game :

This is the main class where all other classes are used in the form of an object to get the desired game running.

Subroutines :

- dispose()
- run()
- moveBall()

Code:

```
 class Game{
  static Game instance;
  field Pad batone;
  field Pad battwo;
  field Ball ball;
  field boolean exit;
  field boolean reset;
  field int score1;
  field int score2;
  field int wall;
  field int batheight;
  field int lastWall;
constructor Game new(){
 do Screen.clearScreen();
let score1=0;
        let score2=0;
   let batheight=50;
   let batone=Pad.new(2,102,10,batheight);
   let battwo=Pad.new(500,102,10,batheight);
   let ball=Ball.new(100,100,12,500,43,255);
  do ball.setDestination(0,200);
  do Screen.drawRectangle(0,40,511,42);
  do Output.moveCursor(2,13);
   do Output.printInt(score1);
   do Output.moveCursor(2,51);
   do Output.printInt(score2);
        do Output.moveCursor(2,28);
```

```
        do Output.printString("( SCORE )");

let exit=false;
let reset=false;

        let lastWall = 0;
        let wall=0;
   return this;
  }
  method void dispose() {
      do batone.dispose();
      do battwo.dispose();
      do Memory.deAlloc(this);
      return;
    }
 function void newInstance() {
      let instance = Game.new();
      return;
    }
  function Game getInstance() {
      return instance;
    }
method void run(){
  var char key;
  var boolean x;
 var boolean fin;
 var char m;
  while (~reset)
```

```
{
        let lastWall = 0;
        let wall=0;
        let exit=false;


  while (~exit) {
        // waits for a key to be pressed.
        while (((key = 0)) & (~exit)) {
           let key = Keyboard.keyPressed();
                                    do batone.move();
                                    do battwo.move();
                                    do moveBall();
           do Sys.wait(50);
        }

        if (key = 87) { do batone.setDirection(1); }
                    else {
              if(key=83){ do batone.setDirection(2);}
                   else{    if(key=131){do battwo.setDirection(1);}
                                        else{


                  if (key = 133) { do battwo.setDirection(2); }
                                        else {
               if (key = 140) { let reset = true; }
                          }}}
        }
```

```
        // Waits for the key to be released...
        while ((~(key = 0)) & (~exit)) {
           let key = Keyboard.keyPressed();
                                    do batone.move();
                                    do battwo.move();
                                    do moveBall();
           do Sys.wait(50);

        }
     }

          if(exit)
                {   if(ball.getLeft() < 200)
                { let score2=score2 + 1;    }
                else
                {  let score1=score1 + 1;}
                do ball.hide();
                do Output.moveCursor(2,13);
        do Output.printInt(score1);
        do Output.moveCursor(2,51);
        do Output.printInt(score2);
                              do ball.dispose();
                             // let batone=Pad.new(2,102,10,batheight);
// let battwo=Pad.new(500,102,10,batheight);
        let ball=Ball.new(200,100,12,500,43,255);
                              do ball.setDestination(0,200);
                    let x = Win();
                              if(x)
```

```
                    {  let reset=true;}

                }
                 }
                   do Output.moveCursor(10,27);
               do Output.printString("GAME OVER");
                        do Sys.wait(1500);
                        if(score1=3)
                        {do Output.moveCursor(12,25);
                        do Output.printString("PLAYER 1 WINS !");}
                        else
                        {
                            do Output.moveCursor(12,25);
                        do Output.printString("PLAYER 2 WINS !");
                        }

                        let fin=true;
                        while(fin)
                        {  do Output.moveCursor(20,11);
                        do Output.printString(" Hold  BACKSPACE  to go back to MAIN
MENU");
                        do Sys.wait(850);
                        let m=Keyboard.keyPressed();
                        if(m=129)
                        { let fin=false; }

                        }
                    return;
                }
```

```
            /**let x = Win();
    if (exit & x) {
 do Output.moveCursor(10,27);
        do Output.printString("Game Over");


    }
           else{
              if((ball.getLeft()) > 400){let score1=score1+1;}
 if((ball.getLeft()) < 20){let score2=score2+1;}
                      do Output.moveCursor(2,21);
  do Output.printInt(score1);
  do Output.moveCursor(2,42);
  do Output.printInt(score2);
do  ball.hide();
               do  ball.setX(100);
              do  ball.setY(100);
              do ball.show();
                      do ball.setDestination(0,200);
                      let exit=false;
                      let lastWall = 0;
 let wall=0;
                      //do Game.new();
 do Sys.wait(50);

 }**/


 method boolean Win()
```

```
        {
          if ((score1=3))
          {  return true;}
          else{
if (score2=3)
{return true;}
        else
        { return false;  }
        }}
        method boolean collision(Pad p)
        { var int batLeft, batRight, ballTop, ballRight,ballLeft,ballBottom,batTop,batBottom;
          let batLeft=p.getLeft();
          let batRight=p.getRight();
          let batTop=p.getTop();
          let batBottom=p.getBottom();
          let ballLeft=ball.getLeft();
          let ballRight=ball.getRight();
          let ballTop=ball.getTop();
          let ballBottom=ball.getBottom();
          if ((ballRight>batLeft) & (ballBottom>batTop) & (ballLeft>batRight) &
(ballTop>batBottom))
        {return true;}
          else{ return false;}


        }
method void moveBall() {
      var int
bouncingDirection,batoneLeft,batoneRight,battwoLeft,battwoRight,ballLeft,ballRight,bonet,btw
ot,boneb,btwob,ballBottom,ballTop;
```

```
let wall = ball.move();


if ((wall > 0) & (~(wall = lastWall))) {
   let lastWall = wall;
   let bouncingDirection = 0;
   let batoneLeft = batone.getLeft();
   let batoneRight = batone.getRight();
   let battwoLeft = battwo.getLeft();
   let battwoRight = battwo.getRight();
                  let ballLeft = ball.getLeft();
   let ballRight = ball.getRight();
                  let bonet=batone.getTop();
                  let btwot=battwo.getTop();
                  let boneb=batone.getBottom();
                  let btwob=battwo.getBottom();
                  let ballBottom=ball.getBottom();
                  let ballTop=ball.getTop();


   if (wall = 1) {
     // let exit = ((bonet > ballBottom) | (boneb < ballTop) | (btwot > ballBottom) |(btwob <
ballTop));
      //let exit = ((batoneLeft+(batone.getWidth)) > ballRight) |
((batoneRight+(batone.getWidth)) > ballLeft)|(battwoLeft > ballRight) | (battwoRight >
ballLeft));
                        let exit=((ballBottom < bonet) | (ballTop > boneb));
                       if (~exit) {
          if (ballBottom > (bonet + 10)) { let bouncingDirection = -1; }
          else {
```

```
            if (ballTop < (boneb - 10)) { let bouncingDirection = 1; }
        }



    }}
                        if(wall=2){
                        let exit=((ballTop>btwob) |(ballBottom<btwot));
                        if (~exit)
                        {
                        if (ballBottom > (btwot + 10)) { let bouncingDirection = -1; }
        else {
            if (ballTop > (btwob - 10)) { let bouncingDirection = 1; }
        }



                        }
                        }



                //let batheight = batheight - 2;
    //    do batone.setWidth(batheight);
    //    do battwo.setWidth(batheight);
            do ball.bounce(bouncingDirection);
    }
    return;
}}
```

Class Main :

In this class, we create an instance of the Game class and decide when to run which screen.

Code:

```
class Main
{
function void main()
{  var Game game;
   var Start s;
   var Start l;
   var char k;
   var boolean st;
   var Controls c;
    let st=true;
    let s= Start.new();
  while (st)
   {let k=Keyboard.keyPressed();

         if(k=129)
           {   do Screen.clearScreen();
do s.dispose();
let s=Start.new();
         }
         if(k=88)
    {
      do Screen.clearScreen();
```

```
    let c= Controls.new();}
  else
       {if(k=32)
  {do Screen.clearScreen();
  do game.newInstance();
  let game=Game.getInstance();
  do game.run();
  do game.dispose();}
 }}
do s.dispose();
do  l.dispose();
do  c.dispose();
  return;
}
}
```

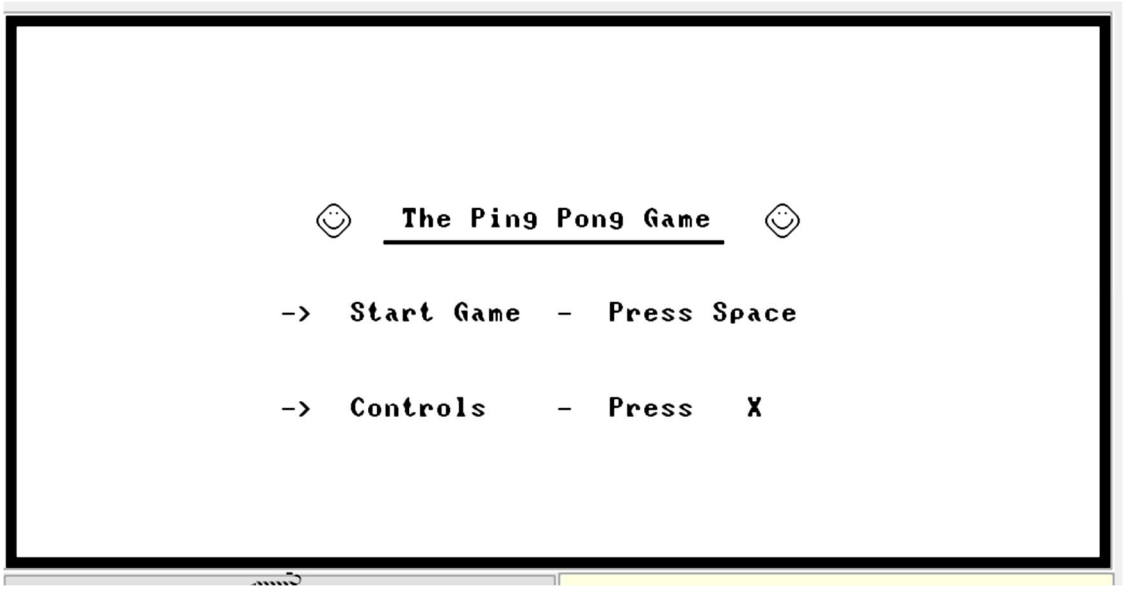
User Manual

1.) At first, we load the Ping Pong folder into the VM Emulator.

2.) Then, we get the welcoming start screen displaying the game name and giving us the two options :

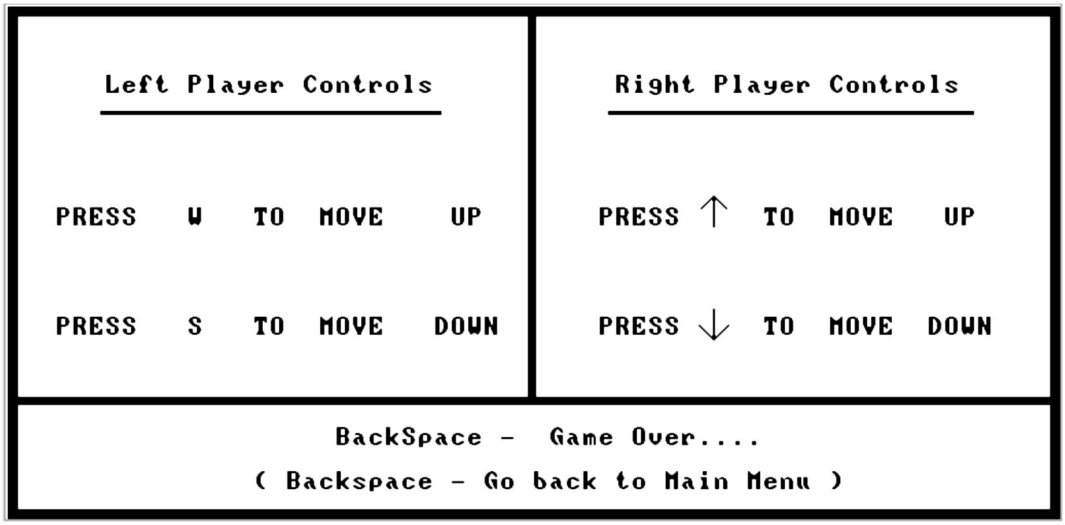
a.) To start the game, the player needs to press Space.

b.) To view the controls screen, the user needs to press X.



3.) Before moving on to the game, at first we need to know what are the controls used to operate the Pad.

So upon pressing X, we get this :

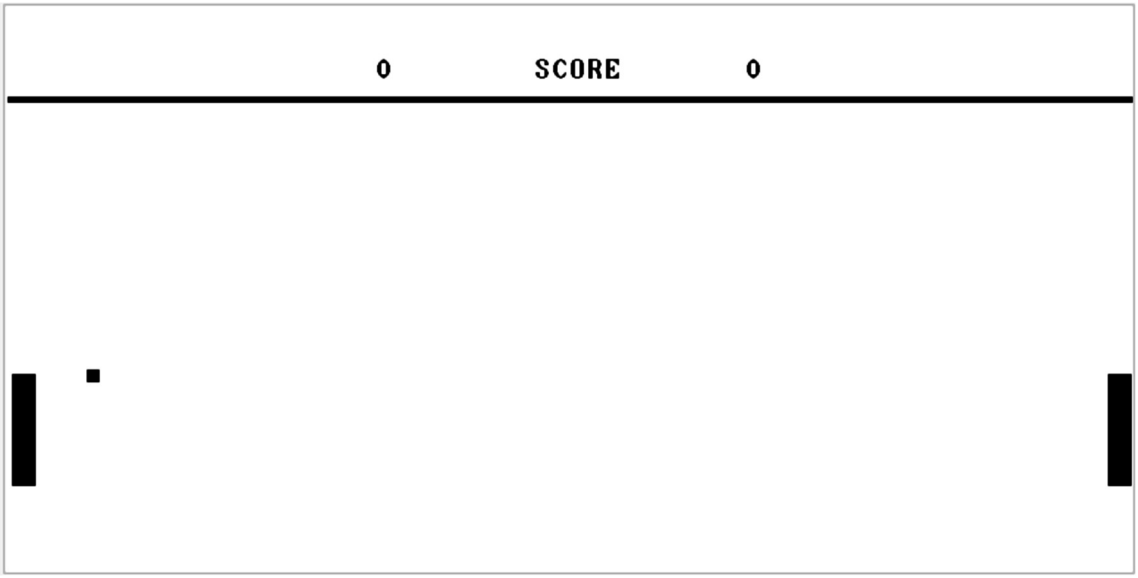


4.) After understanding the controls, the player has to press the Backspace button, to again go back to the start screen to begin the game!

5.) Once, you're on the start screen, press Space to start the game.

6.) Then when the game begins, both the players will be playing the game.

To win the game, any one of the players needs to score 3 points!

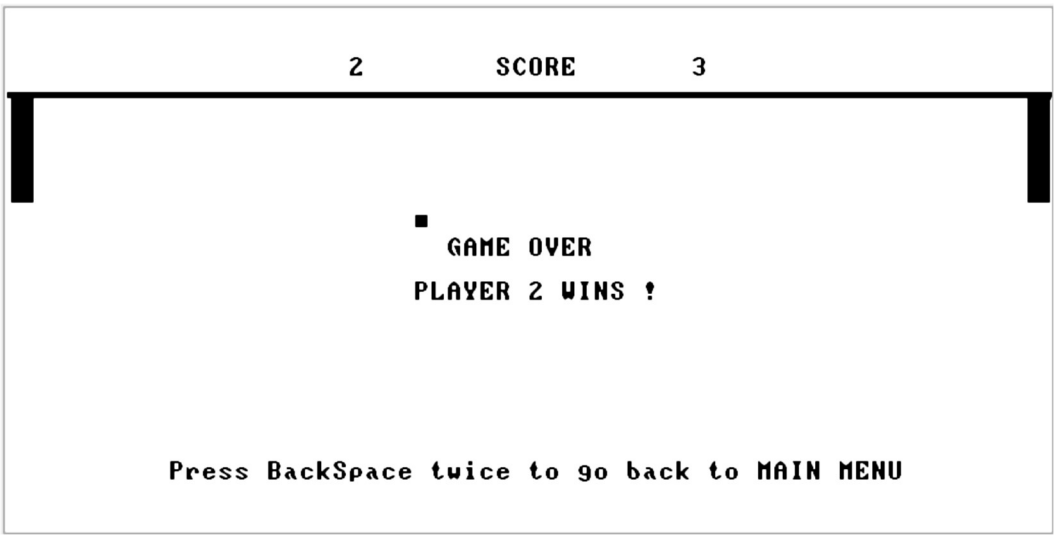


7.) The left player (W/S) will be controlling the left Pad – in the vertical direction,
While the right player (UpArrow/DownArrow) will be controlling the right Pad.

8.) Whoever misses hitting the ball with the Pad, their respective opponent gets an increment in the score by 1.

9.) The first player to score 3 points wins the game. In case if the left player wins, the winning screen looks like this.

And if the right player wins he gets to see something like this :



10.) That's all for you to know, enjoy the game!