**import** acm.graphics.\*;

**import** acm.program.\*;

**import** java.awt.\*;

**import** java.awt.event.MouseEvent;

**public** **class** BrickBreaker **extends** GraphicsProgram{

**public** **static** **final** **int** ***APPLICATION\_WIDTH*** = 400;

**public** **static** **final** **int** ***APPLICATION\_HEIGHT*** = 600;

/\*\* Dimensions of game board. On some platforms these may NOT actually

\* be the dimensions of the graphics canvas. \*/

**private** **static** **final** **int** ***WIDTH*** = ***APPLICATION\_WIDTH***;

**private** **static** **final** **int** ***HEIGHT*** = ***APPLICATION\_HEIGHT***;

/\*\* Dimensions of the paddle \*/

**private** **static** **final** **int** ***PADDLE\_WIDTH*** = 60;

**private** **static** **final** **int** ***PADDLE\_HEIGHT*** = 10;

/\*\* Offset of the paddle up from the bottom \*/

**private** **static** **final** **int** ***PADDLE\_Y\_OFFSET*** = 30;

/\*\* Number of bricks per row \*/

**private** **static** **final** **int** ***NBRICKS\_PER\_ROW*** = 10;

/\*\* Number of rows of bricks \*/

**private** **static** **final** **int** ***NBRICK\_ROWS*** = 10;

/\*\* Separation between bricks \*/

**private** **static** **final** **int** ***BRICK\_SEP*** = 4;

/\*\* Width of a brick \*/

**private** **static** **final** **int** ***BRICK\_WIDTH*** =

(***WIDTH*** - (***NBRICKS\_PER\_ROW*** - 1) \* ***BRICK\_SEP***) / ***NBRICKS\_PER\_ROW***;

/\*\* Height of a brick \*/

**private** **static** **final** **int** ***BRICK\_HEIGHT*** = 8;

/\*\* Radius of the ball in pixels \*/

**private** **static** **final** **int** ***BALL\_RADIUS*** = 10;

/\*\* Offset of the top brick row from the top \*/

**private** **static** **final** **int** ***BRICK\_Y\_OFFSET*** = 70;

**boolean** IsRunning = **false**;

/\*\* Number of turns \*/

**private** **static** **final** **int** ***NTURNS*** = 3;

**private** GOval circle;

**private** GRect paddle;

**private** GRect background;

**private** GRect[] Bricks = **new** GRect[***NBRICK\_ROWS*** \* ***NBRICKS\_PER\_ROW***];

**private** **boolean**[] VisibleBricks = **new** **boolean**[***NBRICK\_ROWS*** \* ***NBRICKS\_PER\_ROW***];

**public** **void** run() {

**int** width = getWidth();

**int** height = getHeight();

// Setup function here

// Reset bricks,boolean values,

**if**(IsRunning == **false**){

setupFunction();

IsRunning = **true**;

}

// GRect brick = new GRect ();

**int** dx = 3;

**int** dy = 3;

waitForClick();

**while**(**true**){

circle.move(dx, dy);

pause(20);

**double** getX = circle.getX();

**double** getY = circle.getY();

**if** (getX <= 0 ){ // when hits walls, reset to middle

dx = -dx;

}

**if** (getY <= 0){

dy = -dy;

}

**if** (getX + ***BALL\_RADIUS*** \* 2 >= width){

dx = -dx;

}

**if** (getY + ***BALL\_RADIUS*** \* 2 >= height){

// position ball at middle of screen

**double** px1 = paddle.getX();

**double** px2 = paddle.getX() + ***PADDLE\_WIDTH***;

**if**(getX-***BALL\_RADIUS*** <= px2 && getX+***BALL\_RADIUS*** >= px1){

dy \*= -1;

}

**else**{

circle.setLocation(width / 2 - ***BALL\_RADIUS***, height / 2 - ***BALL\_RADIUS***);

}

//dy = -dy;

}

**if**(changeDirection()== **true**){

dy \*= -1;

// if our x value of ball + radius = bricks getX

//dx\*= -1

}

}

}

**private** **boolean** dectectOverlap(**double** x,**double** y){

**double** CX = circle.getX();

**double** CY = circle.getY();

**if**(CX-***BALL\_RADIUS*** >= x && CX+***BALL\_RADIUS*** <= x + ***BRICK\_WIDTH*** ){

**if**(CY-***BALL\_RADIUS*** <= y && CY+***BALL\_RADIUS*** <= y + ***BRICK\_HEIGHT***){

**return** **true**;}

}

**return** **false**;

}

**private** **boolean** changeDirection(){

**for**(**int** i = 0; i < Bricks.length; i++ ){

**boolean** Visible = VisibleBricks[i];

**double** lbx = Bricks[i].getX();

**double** lby = Bricks[i].getY();

**if**(Visible == **true** && dectectOverlap(lbx, lby)== **true**){

VisibleBricks[i] = **false**;

Bricks[i].setFilled(**false**);

**return** **true**;

}

}

**return** **false**;

}

**private** **void** setupFunction(){

**int** xpos = 0;

**int** ypos = 0;

**int** i,j = 0;

**for** (i = 0; i < ***NBRICKS\_PER\_ROW***; i++){

xpos = (***BRICK\_WIDTH*** + ***BRICK\_SEP***) \* i;

**for** (j = 0; j < ***NBRICK\_ROWS***; j++){

**int** index = (i \* 10) + j;

VisibleBricks[index]= **true**;

ypos = (***BRICK\_HEIGHT*** + ***BRICK\_SEP***) \* j + ***BRICK\_Y\_OFFSET***;

Bricks[index] = **new** GRect(xpos,ypos,***BRICK\_WIDTH***,***BRICK\_HEIGHT***);

add(Bricks[index]);

Bricks[index].setColor(Color.***DARK\_GRAY***);

Bricks[index].setFilled(**true**);

}

}

**int** width = getWidth();

**int** height = getHeight();

circle = **new** GOval (width / 2 - ***BALL\_RADIUS***, height / 2 - ***BALL\_RADIUS***, ***BALL\_RADIUS*** \* 2, ***BALL\_RADIUS*** \* 2);

circle.setColor(Color.***RED***);

circle.setFilled(**true**);

add(circle);

paddle = **new** GRect (width / 2 - ***PADDLE\_WIDTH*** / 2, height - ***PADDLE\_Y\_OFFSET***, ***PADDLE\_WIDTH***, ***PADDLE\_HEIGHT*** );

add(paddle);

paddle.setColor(Color.***BLACK***);

paddle.setFilled(**true**);

addMouseListeners();

background = **new** GRect (***WIDTH***, ***HEIGHT***,0,0 );

background.setColor(Color.***LIGHT\_GRAY***);

background.setFilled(**true**);

add(background);

}

**public** **void** Collision(){

}

**public** **void** mouseMoved(MouseEvent e){

e.getX();

paddle.setLocation(e.getX(), ***HEIGHT*** - ***PADDLE\_Y\_OFFSET***);

// e.getX will get value of x pos of mouse

// if mouse x is bigger than board width then set paddle x to board width

// it stays the same

}

}