```
//on the lab description, explain how we arrived at the document width, height (i chose 16x9)
//explain how we arrived at the game container width,height and box dimensions

//game container is 320 pixels wide. instance name: gameContainer
//with 10 boxes across, that's a 32x32 box
//game container is 448 pixels high, so 14 rows at 32 pixels per row
//nextBlock container is...

//create red block in photoshop. 32x32. radial fill with red/black. blending>inner glow with white at
//   export as png8, no transparency, no interlacing.
//   then, new symbol, file>import to stage... choose the png
//   repeat for other colors of block

//After creating all 7 block types,
//   right click on each in the library, advanced, export for actionscript, should see class name unde

//add background layer - create background for game

//create labels and textboxes for score, lines, and level
//name the textboxes, read only: score, lines, level

//prelab
//make all the assets - follow my instructions or deviate slightly for your own effects


//1. declare and initialize variables
//2. randomly select a block and place in on game board
//3. make the keyboard arrows work
//4. collision detection with the walls of the gameContainer
//5. set the game piece
//6.
//make the block fall
//collision detection
//scoring

import flash.display.Bitmap;
import flash.display.BitmapData;
import Key;
import flash.utils.getTimer;
import flash.geom.Matrix;
import flash.geom.Point;
import flash.geom.Rectangle;

Key.initialize(stage);
this.addEventListener('enterFrame', keyPressed);                        //event listener to catch key p

var startTime = flash.utils.getTimer();                                 //Milliseconds since flash play
var lastTimeKeyPressed = flash.utils.getTimer() - startTime;            //last time a key was pressed
var millisecondsElapsed = flash.utils.getTimer() - lastTimeKeyPressed; //time elapsed between key pres
var nextBlockRandomNum:int = 0;
var currentBlockRandomNum:int = 0;
var currentBlock = null;
var nextBlock = null;
var inGameContainer:Boolean = true;                                      //boolean for whether next mov
var allBlocks:AllBlocks = new AllBlocks();                               //container for finished block
var numRotations:Number = 0;

//bitmap data... true, 0x00000000 are required in order for it to be transparent
var bitmapGameData:BitmapData  = new BitmapData(320, 448, true, 0x00000000);    //all the game pieces
var bitmapGameContainer:Bitmap = new Bitmap(bitmapGameData);

//declare and initialize variables
var blockArray:Array = new Array(0,1,2,3,4,5,6);
      //yours may not be exactly the same as mine
      //0 = Purple Reverse Z
      //1 = Yellow Z
      //2 = Silver Square
      //3 = Orange Line
      //4 = Red L
      //5 = Green L
      //6 = Blue T
      //trace(blockArray);
```

```
//set game container bounds
gameContainer.x      = 144;
gameContainer.y      = 24;
gameContainer.width  = 320;
gameContainer.height = 448;
//set next block container bounds
nextBlockContainer.x      = 490;
nextBlockContainer.y      = 24;
nextBlockContainer.width  = 160;
nextBlockContainer.height = 80;
//set the bitmap container bounds
bitmapGameContainer.x      = 144;
bitmapGameContainer.y      = 24;
bitmapGameContainer.width  = 320;
bitmapGameContainer.height = 448;

stage.addChild(allBlocks);
stage.addChild(bitmapGameContainer);

initialize();

//initialize the game
function initialize():void
{
    getNextBlock();     //first time only sets the next block container
    getNextBlock();     //second time sets a piece in game container and next block container
}


////////////////////////////////////////////////////
// function getNextBlock
// randomly select a block and place it on game board
////////////////////////////////////////////////////
function getNextBlock():void
{
    if(nextBlock != null)
    {
        currentBlock = nextBlock;                       //set currentBlock equal to the nextBlock
        currentBlockRandomNum = nextBlockRandomNum;     //save random num for the current block to use
        nextBlockContainer.removeChild(nextBlock);      //empty the nextBlock container
        stage.addChild(currentBlock);                   //add new currentBlock to the stage

        currentBlock.x = currentBlock.x - currentBlock.getBounds(stage).x + 240;    //position block o
        currentBlock.y = currentBlock.y - currentBlock.getBounds(stage).y + 24;     //position block o
    } //end if

    nextBlockRandomNum = Math.floor(Math.random() * blockArray.length);     //randomly select one of 7

    numRotations = 0;                                   //reset number of rotations

    //uncomment to test a specific block
    nextBlockRandomNum = 0;

    //call buildNextBlock
    //buildNextBlock(nextBlockRandomNum);

    if(nextBlockRandomNum == 0)
        nextBlock = new Purple5();
    else if(nextBlockRandomNum == 1)
        nextBlock = new YellowZ();
    else if(nextBlockRandomNum == 2)
        nextBlock = new SilverSquare();
    else if(nextBlockRandomNum == 3)
        nextBlock = new OrangeLine();
    else if(nextBlockRandomNum == 4)
        nextBlock = new RedL();
    else if(nextBlockRandomNum == 5)
        nextBlock = new GreenL();
    else if(nextBlockRandomNum == 6)
        nextBlock = new BlueT();
```

```
    nextBlockContainer.addChild(nextBlock);                                 //add new nextBlock to the nex

    nextBlock.x = nextBlock.x - nextBlock.getBounds(nextBlock).x + ((160-nextBlock.width)/2);   //cent
    nextBlock.y = nextBlock.y - nextBlock.getBounds(nextBlock).y + ((80-nextBlock.height)/2);   //cent
}
//////////////////////////////////////////////////////////
// end getNextBlock
//////////////////////////////////////////////////////////



//////////////////////////////////////////////////////////
// function keyPressed
// This function is called constantly by the stage
// Each time it is called, we check to see how much time
// has elapsed since the last time in order to prevent
// a rapid fire by holding down the arrow key.
// After that, it's simply a matter of moving or rotating
// the piece based on which arrow was pressed.
//////////////////////////////////////////////////////////
function keyPressed(e:Event):void
{
    //Check for arrow keys so that we can grab the time elapsed
    if(Key.isDown(Keyboard.LEFT) || Key.isDown(Keyboard.RIGHT) || Key.isDown(Keyboard.UP) || Key.isDow
    {
        millisecondsElapsed = flash.utils.getTimer() - lastTimeKeyPressed;  //set the time elapsed bet
        //trace("last: " + lastTimeKeyPressed + " milli: " + millisecondsElapsed);
        lastTimeKeyPressed = flash.utils.getTimer();                        //set the last time the ke
    }

    if(Key.isDown(Keyboard.LEFT) && (millisecondsElapsed > 100))            //check for left arrow and
    {
        if((!extendsOutsideGameContainer("left")) && (!collidesLeft()))
            currentBlock.x -= 32;                                           //move the piece to the le
    }
    else if(Key.isDown(Keyboard.RIGHT) && (millisecondsElapsed > 100))      //check for right arrow an
    {
        if((!extendsOutsideGameContainer("right")) && (!collidesRight()))
            currentBlock.x += 32;                                           //move the piece to the ri
    }
    else if(Key.isDown(Keyboard.DOWN) && (millisecondsElapsed > 100))       //check for down arrow and
    {
        if((!extendsOutsideGameContainer("down")) && (!collidesDown()))
            currentBlock.y += 32;                                           //move the piece down
        else
            setGamePiece();
    }
    else if(Key.isDown(Keyboard.UP) && (millisecondsElapsed > 100))         //check for up arrow and t
    {
        currentBlock.rotation += 90;
        if(extendsOutsideGameContainer("up") || (collidesWhenRotated()))
            currentBlock.rotation -= 90;                                    //collides, rotate the pie
        else
            numRotations++;
    }

} //end function keyPressed
//////////////////////////////////////////////////////////
// end keyPressed
//////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////
// function extendsOutsideGameContainer
// Parameters: accepts one parameter, a string, which
// represents the direction arrow that the user pressed.
// Returns: a boolean, true or false, as to whether the
// piece is allowed to move in that direction. True means
// it extends outside the game container boundaries: do not
// move/rotate. False means it does not extend outside the
// game container boundaries: OK to move/rotate.
// Purpose: Checks only for containment within the playing
```

```
// area so that the game piece does not move outside the
// playing area. Other collision detection is done
// elsewhere.
//////////////////////////////////////////////////////
function extendsOutsideGameContainer(direct:String):Boolean
{
    // 24 is gameContainer y position on stage
    //144 is gameContainer x position on stage
    //464 is gameContainer x + width + currentBlock width
    //496 is gameContainer y + height + currentBlock height
    if((direct == "left") && (currentBlock.getBounds(stage).x - 32 < 144))
        return true; //the piece is NOT allowed to move left
    else if((direct == "right") && (currentBlock.getBounds(stage).x + currentBlock.width + 32 > 464))
        return true; //the piece is NOT allowed to move right
    else if((direct == "down") && (currentBlock.getBounds(stage).y + currentBlock.height + 32 > 496))
        return true; //the piece is NOT allowed to move down
    else if(direct == "up")
    {
        if(currentBlock.getBounds(stage).y < 24)
            return true; //piece would be outside the top border, rotate back
        else if(currentBlock.getBounds(stage).x < 144)
            return true; //piece would be outside the left border, rotate back
        else if(currentBlock.getBounds(stage).x + currentBlock.width > 464)
            return true; //piece would be outside the right border, rotate back
        else if(currentBlock.getBounds(stage).y + currentBlock.height > 496)
            return true; //piece would be outside the bottom border, rotate back
        else
            return false; //OK to rotate piece
    }
    else
        return false;  //the piece is allowed to move that direction
}
//////////////////////////////////////////////////////
// end extendsOutsideGameContainer
//////////////////////////////////////////////////////


function setGamePiece():void
{
    allBlocks.addChild(currentBlock);
    var myMatrix:Matrix = new Matrix();                     //create a new matrix
    myMatrix.rotate(((numRotations%4)*90)*Math.PI/180);     //rotate the piece, must rotate before tra
    myMatrix.tx = (currentBlock.x - bitmapGameContainer.x); //translate on x axis
    myMatrix.ty = (currentBlock.y - bitmapGameContainer.y); //translate on y axis
//  trace("radians: " + (((numRotations%4)*90)*Math.PI/180));
//  trace("tx: " + translateX);
//  trace("ty: " + translateY);
//  trace("cb x: " + currentBlock.x);
//  trace("cb y: " + currentBlock.y);
//  trace("all x: " + allBlocks.x);
//  trace("all y: " + allBlocks.y);

    bitmapGameData.draw(currentBlock, myMatrix);


    getNextBlock();
}



//////////////////////////////////////////////////////
// function collidesWhenRotated
//////////////////////////////////////////////////////
function collidesWhenRotated():Boolean
{
    //degrade gracefully - this line only kinda works and should never execute if coded properly
    return allBlocks.hitTestObject(currentBlock);
}
//////////////////////////////////////////////////////
// function collidesWhenRotated
//////////////////////////////////////////////////////
```

```
function collidesDown():Boolean
{
    var point1:Point = new Point(0,0);
    var point2:Point;
    var point3:Point;
    var point4:Point;
    var point5:Point;

    if(currentBlockRandomNum == 0)
    {   //Purple Reverse Z
        if(numRotations%4 == 0)
        {   //three blocks horizontal, reg mark one block from right
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
            point4 = new Point(currentBlock.x-bitmapGameContainer.x-48, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 2)
        {   //three blocks horizontal, reg mark two blocks from right
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
            point4 = new Point(currentBlock.x-bitmapGameContainer.x+48, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 1)
        {   //three blocks vertical, reg mark one block from bottom
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 3)
        {   //three blocks vertical, reg mark two blocks from bottom
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
        }//end if...else

        //run the hitTests
        if(numRotations%2 == 0)
        {   //even number of rotations
            if(bitmapGameData.hitTest(point1, 0xFF, point2))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point3))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point4))
                return true;
            else
                return false;
        }
        else
        {   //odd number of rotations
            if(bitmapGameData.hitTest(point1, 0xFF, point2))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point3))
                return true;
            else
                return false;
        }//end if...else

        return false;
    }
    else if(currentBlockRandomNum == 1)
    {   //Yellow Z
    }
    else if(currentBlockRandomNum == 2)
    {   //silver square
        point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameContainer
        point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameContainer

        if(bitmapGameData.hitTest(point1, 0xFF, point2))
            return true;
        else if(bitmapGameData.hitTest(point1, 0xFF, point3))
            return true;
```

```
        else
            return false;
    }
    else if(currentBlockRandomNum == 3)
    {   //orange line
        if(numRotations%4 == 0)
        {   //horizontal with registration on top
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
            point4 = new Point(currentBlock.x-bitmapGameContainer.x-48, currentBlock.y-bitmapGameConta
            point5 = new Point(currentBlock.x-bitmapGameContainer.x+48, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 2)
        {   //horizontal with registration on bottom
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
            point3 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
            point4 = new Point(currentBlock.x-bitmapGameContainer.x-48, currentBlock.y-bitmapGameConta
            point5 = new Point(currentBlock.x-bitmapGameContainer.x+48, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 1)
        {   //vertical with registration on right
            point2 = new Point(currentBlock.x-bitmapGameContainer.x-16, currentBlock.y-bitmapGameConta
        }
        else if(numRotations%4 == 3)
        {   //horizontal with registration on left
            point2 = new Point(currentBlock.x-bitmapGameContainer.x+16, currentBlock.y-bitmapGameConta
        }//end if...else

        //run the hitTests
        if(numRotations%2 == 0)
        {   //laying horizontal
            if(bitmapGameData.hitTest(point1, 0xFF, point2))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point3))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point4))
                return true;
            else if(bitmapGameData.hitTest(point1, 0xFF, point5))
                return true;
            else
                return false;
        }
        else
        {   //laying vertical
            if(bitmapGameData.hitTest(point1, 0xFF, point2))
                return true;
            else
                return false;
        }//end if...else


        return false;
    }
    else if(currentBlockRandomNum == 4)
    {   //Red L
    }
    else if(currentBlockRandomNum == 5)
    {   //Green L
    }
    else if(currentBlockRandomNum == 6)
    {   //Blue T
    }

    //degrade gracefully - this line only kinda works and should never execute if coded properly
    //return bitmapGameContainer.hitTestPoint(currentBlock.x, currentBlock.y, true);
    return false;
}


function collidesRight():Boolean
{
    var point1:Point = new Point(1,1);
```

```
    if(currentBlockRandomNum == 0)
    {   //Purple Reverse Z
    }
    else if(currentBlockRandomNum == 1)
    {   //Yellow Z
    }
    else if(currentBlockRandomNum == 2)
    {   //silver square
        var point2:Point = new Point(currentBlock.x-bitmapGameContainer.x+64, currentBlock.y-bitmapGam
        var point3:Point = new Point(currentBlock.x-bitmapGameContainer.x+64, currentBlock.y-bitmapGam

        if(bitmapGameData.hitTest(point1, 0xFF, point2))
            return true;
        else if(bitmapGameData.hitTest(point1, 0xFF, point3))
            return true;
        else
            return false;
    }
    else if(currentBlockRandomNum == 3)
    {   //orange line
    }
    else if(currentBlockRandomNum == 4)
    {   //Red L
    }
    else if(currentBlockRandomNum == 5)
    {   //Green L
    }
    else if(currentBlockRandomNum == 6)
    {   //Blue T
    }

    //degrade gracefully - this line only kinda works and should never execute if coded properly
    //return bitmapGameContainer.hitTestPoint(currentBlock.x, currentBlock.y, true);
    return false;
}


function collidesLeft():Boolean
{
    var point1:Point = new Point(bitmapGameContainer.x, bitmapGameContainer.y);

//var redRect:Rectangle = redClip.getBounds(this);
//var redClipBmpData = new BitmapData(redRect.width, redRect.height, true, 0);
//redClipBmpData.draw(redClip);

//var blueRect:Rectangle = blueClip.getBounds(this);
//var blueClipBmpData = new BitmapData(blueRect.width, blueRect.height, true, 0);
//blueClipBmpData.draw(blueClip);


var gamePieceBmpData:BitmapData = new BitmapData(currentBlock.width, currentBlock.height, true, 0);
gamePieceBmpData.draw(currentBlock);

//redClipBmpData.hitTest(new Point(redClip.x, redClip.y),
//                       255,
//                       blueClipBmpData,
//                       new Point(blueClip.x, blueClip.y),
//                       255)

    if(currentBlockRandomNum == 0)
    {   //Purple Reverse Z
    }
    else if(currentBlockRandomNum == 1)
    {   //Yellow Z
    }
    else if(currentBlockRandomNum == 2)
    {   //silver square
        var point2:Point = new Point(currentBlock.x-bitmapGameContainer.x-32, currentBlock.y-bitmapGam
        var point3:Point = new Point(currentBlock.x-bitmapGameContainer.x-32, currentBlock.y-bitmapGam

        if(bitmapGameData.hitTest(point1, 255, gamePieceBmpData, new Point(currentBlock.x, currentBloc
```

```
            return true;
//      else if(bitmapGameData.hitTest(point1, 0xFF, point3))
//          return true;
        else
            return false;
    }
    else if(currentBlockRandomNum == 3)
    {   //orange line
    }
    else if(currentBlockRandomNum == 4)
    {   //Red L
    }
    else if(currentBlockRandomNum == 5)
    {   //Green L
    }
    else if(currentBlockRandomNum == 6)
    {   //Blue T
    }

    //degrade gracefully - this line only kinda works and should never execute if coded properly
    //return bitmapGameContainer.hitTestPoint(currentBlock.x, currentBlock.y, true);
    return false;
}

//////////////////////////////////////////////////////
// function collidesWithAllBlocks
// This function only cares about the bottom surface of
// the game piece. It checks to see if the bottom surface
// of any of the individual 32x32 blocks is colliding with
// a piece that is already in allBlocks.
// Be very mindful of your + and - symbols. Also be very
// mindful of the values you're adding or subtracting.
// It would be very easy to mess up the collision
// detection in this function.
//////////////////////////////////////////////////////
function collidesWithAllBlocks():Boolean
{
    if(currentBlockRandomNum == 0)
    {   //Purple Reverse Z
        if(numRotations%4 == 0)
        {   //three blocks horizontal, reg mark one block from right
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y+32, true))
                return true;
        }
        else if(numRotations%4 == 2)
        {   //three blocks horizontal, reg mark two blocks from right
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y, true))
                return true;
        }
        else if(numRotations%4 == 1)
        {   //three blocks vertical, reg mark one block from bottom
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
        }
        else if(numRotations%4 == 3)
        {   //three blocks vertical, reg mark two blocks from bottom
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+64, true))
                return true;
        }
```

```
    return false;
}
else if(currentBlockRandomNum == 1)
{   //Yellow Z
    if(numRotations%4 == 0)
    {   //three blocks horizontal, reg mark one block from right
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y, true))
            return true;
    }
    else if(numRotations%4 == 2)
    {   //three blocks horizontal, reg mark two blocks from right
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y+32, true))
            return true;
    }
    else if(numRotations%4 == 1)
    {   //three blocks vertical, reg mark one block from bottom
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
            return true;
    }
    else if(numRotations%4 == 3)
    {   //three blocks vertical, reg mark two blocks from bottom
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+64, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
    }

    return false;
}
else if(currentBlockRandomNum == 2)
{   //silver square
    if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
        return true;
    else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
        return true;
    else
        return false;
}
else if(currentBlockRandomNum == 3)
{   //orange line
    //originally tried using this. Did not work because getBounds didn't change after rotation.
    //if((boundsY == 0) && (boundsX < boundsY))
    //else if((boundsY < 0) && (boundsX < boundsY))
    //else if((boundsX < 0) && (boundsX > boundsY))
    //else if((boundsX == 0) && (boundsX > boundsY))

    if(numRotations%4 == 0)
    {   //horizontal with registration on top
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y+32, true))
            return true;
    }
    else if(numRotations%4 == 2)
    {   //horizontal with registration on bottom
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
```

```
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y, true))
            return true;
    }
    else if(numRotations%4 == 1)
    {   //vertical with registration on right
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+64, true))
            return true;
    }
    else if(numRotations%4 == 3)
    {   //horizontal with registration on left
        if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+64, true))
            return true;
    }

    return false;
}
else if(currentBlockRandomNum == 4)
{   //Red L
    if(numRotations%4 == 0)
    {   //three blocks on bottom
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y+32, true))
            return true;
    }
    else if(numRotations%4 == 2)
    {   //three blocks on top
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y, true))
            return true;
    }
    else if(numRotations%4 == 1)
    {   //three blocks on left
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+64, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
            return true;
    }
    else if(numRotations%4 == 3)
    {   //three blocks on right
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
    }

    return false;
}
else if(currentBlockRandomNum == 5)
{   //Green L
    if(numRotations%4 == 0)
    {   //three blocks on bottom
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y+32, true))
            return true;
    }
    else if(numRotations%4 == 2)
    {   //three blocks on top
        if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
            return true;
        else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
```

```
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y, true))
                return true;
        }
        else if(numRotations%4 == 1)
        {   //three blocks on left
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
        }
        else if(numRotations%4 == 3)
        {   //three blocks on right
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+64, true))
                return true;
        }

        return false;
    }
    else if(currentBlockRandomNum == 6)
    {   //Blue T
        if(numRotations%4 == 0)
        {   //three blocks on bottom
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+48, currentBlock.y+32, true))
                return true;
        }
        else if(numRotations%4 == 2)
        {   //three blocks on top
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+32, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x-48, currentBlock.y, true))
                return true;
        }
        else if(numRotations%4 == 1)
        {   //three blocks on left
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y+64, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
        }
        else if(numRotations%4 == 3)
        {   //three blocks on right
            if(allBlocks.hitTestPoint(currentBlock.x-16, currentBlock.y, true))
                return true;
            else if(allBlocks.hitTestPoint(currentBlock.x+16, currentBlock.y+32, true))
                return true;
        }

        return false;
    }

    //degrade gracefully - this line only kinda works and should never execute if coded properly
    return allBlocks.hitTestPoint(currentBlock.x, currentBlock.y, true);
}
//////////////////////////////////////////////////////
// end function collidesWithAllBlocks
//////////////////////////////////////////////////////
```