



Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Software Development for Android Devices

ENCS 539

Project #5

Flying Flashcards

Student Name: Mohammad Kabajah

ID: 1110600

Instructor's Name: Stephen Taylor

Section #:1

Date: 03/12/2014

Features of the game

Flying flashcards game, from its name the cards is flying from the top of the screen to the bottom of the screen, each card containing an equation in side, after the card hit the bottom or hit another card beneath in the card change the equation with the value of the result. If the two cards with the same result value the two cards will disappear and the score will increase. You can move the cards using two buttons to the left or to the right.

While playing if there is too many cards above each other and the card is on one of the sides you can't move it to the other side of them if they are higher than the moving card, and while playing if there is too many cards above each other and they reach to the top of the screen and another card come from that side you will lose the game, you can always start new game by clicking the new game button if you lose or even in the middle of the game if you don't want to see You lose in the middle of the screen.

Here is the source code, followed by the screenshots of the game while running and all the files will be added in the archive.

```
package edu.birzeit.fall2014.encs539.id1110600.flyingflashcards;

import java.util.Random;
import android.graphics.Color;
import android.graphics.Paint;

public class Cell {

    /**
     * each cell contain a String for the equation a paint object for the
    colour
     * of the card and a result for the equation
     *
     * **/
    String s = "";
    Paint paint = new Paint();
    int result;

    /**
     * create each cell and generate the colour of the card randomly the +128
    in
     * int b = rand.nextInt(128) + 128; is added in case all the randomly
     * generated numbers is 0 so the card will be black and we can't read the
     * equation inside
     *
     * **/
    public Cell() {
        Random rand = new Random();
        int r = rand.nextInt(255);
        int g = rand.nextInt(255);
        int b = rand.nextInt(128) + 128;
        int color = Color.rgb(r, g, b);
        paint.setColor(color);
    }
}
```

```

package edu.birzeit.fall2014.encs539.id1110600.flyingflashcards;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;

public class BoardView extends View {
    public BoardView(Context context) {
        super(context);
        init();
    }

    public BoardView(Context context, AttributeSet as) {
        super(context, as);
        init();
    }

    public BoardView(Context context, AttributeSet as, int t) {
        super(context, as, t);
        init();
    }

    Paint paint = new Paint();
    final static int rows = 10;
    final static int columns = 5;
    Cell[][] cells = new Cell[rows][columns];
    Rect rect = new Rect();
    Rect rect2 = new Rect();
    boolean isLost = false;

    @Override
    public void onDraw(Canvas canvas) {
        int h = getHeight();
        int w = getWidth();
        float dcell_hor = (float) w / columns;
        float dcell_ver = (float) h / rows;

        canvas.drawLine(0, h, w, h, paint);
        canvas.drawLine(0, 1, w, 1, paint);
        canvas.drawLine(w - 1, 0, w - 1, h, paint);
        canvas.drawLine(1, 0, 1, h, paint);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (!cells[i][j].s.equals("")) {
                    rect.top = (int) (i * dcell_ver);
                    rect.left = (int) (j * dcell_hor);
                    rect.right = rect.left + (int) dcell_hor;
                    rect.bottom = rect.top + (int) dcell_ver;
                    canvas.drawRect(rect, cells[i][j].paint);
                    int size = R.dimen.textSize;
                }
            }
        }
    }
}

```

```

        String text = cells[i][j].s;
        paint.setTextSize(getResources()
            .getDimensionPixelSize(size));
        cells[i][j].paint.getTextBounds(text, 0,
text.length(),
            rect);
        canvas.drawText(cells[i][j].s, (rect.right -
rect.width()
            / 2 + (int) (dcell_hor * (j + .25)),
            (rect.top - rect.bottom) / 2
                + (int) (dcell_ver * (i
+ .75))), paint);
    }
}
}
if (isLost) {
    int size = R.dimen.textSize2;

    String text = "You Lose";
    paint.setTextSize(getResources().getDimensionPixelSize(size));
    paint.getTextBounds(text, 0, text.length(), rect2);
    paint.setColor(Color.RED);
    canvas.drawText("You lose", (w - rect2.width()) / 2, h / 2,
paint);
}

}

/**
 * set the position of the cell with the indexes provided
 *
 * **/
public void setPosition(Cell c, int ro, int co) {
    cells[ro][co] = c;
}

/**
 * initialise the array that will hold all the cards
 *
 * **/
public void init() {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cells[i][j] = new Cell();
        }
    }
}

/**
 * check if the second cell with the indexes r2 and c2 is available and set
 * the cell in it and remove it from the old cell
 *
 * **/
public boolean check(int r1, int c1, int r2, int c2) {
    boolean flag = false;

    if (cells[r2][c2].s.equals("")) {
        cells[r2][c2].s = cells[r1][c1].s + "";
    }
}

```

```

        cells[r2][c2].paint = cells[r1][c1].paint;
        cells[r1][c1] = new Cell();
        flag = true;
    }
    return flag;
}

/**
 * check if the next position of the cell is available or not so we will
 * flip the card or not
 *
 * **/
public boolean checkFlip(int r1, int c1, int r2, int c2) {
    boolean flag = false;

    try {
        if (!cells[r2][c2].s.equals("")) {
            flag = true;
        }
    } catch (ArrayIndexOutOfBoundsException e) {
    }
    return flag;
}

/**
 * check to see if the column we want to move to in this row is free or not
 * so we can move it across or skip the cards if there is a card in the
same
 * row in the next column so the card will not overlap
 *
 * if there is in the middle column too many cards and the card on the left
 * or the right side it cant move to the other side if there is some card
in
 * the middle of them
 *
 * **/
public boolean checkEmpty(int r1, int c1, int r2, int c2) {
    boolean flag = false;
    if (cells[r2][c2].s.equals("")) {
        flag = true;
    }
    return flag;
}

/**
 * check if the result of the two cards is the same then we will remove
 * these two cards
 *
 * **/
public boolean checkResults(int ro, int co) {
    boolean flag = false;
    if (ro <= 8) {
        int s1 = cells[ro][co].result;
        int s2 = cells[ro + 1][co].result;
        if (s1 == s2) {
            cells[ro][co] = new Cell();
            cells[ro + 1][co] = new Cell();
            flag = true;
        }
    }
}

```

```

        }

        return flag;
    }

    /**
     * check if the cell in the first row and if the column is full so there is
     * no available space for this cell to be in this column and return true to
     * end the game
     *
     * **/
    public boolean checkGameEnd(int ro, int co) {
        boolean flag = false;
        if (!cells[ro][co].s.equals("") && ro == 0)
            flag = true;
        return flag;
    }
}

package edu.birzeit.fall2014.encs539.id1110600.flyingflashcards;

import java.util.Random;
import android.support.v7.app.ActionBarActivity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    BoardView boardview;
    TextView tv; // textview for the score
    Button left;
    Button right;
    Button newGame;
    Random random = new Random();
    Thread t;
    int row, r2;
    int col, c2;
    Cell c;
    int score = 0;
    boolean threadnull = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        boardview = (BoardView) findViewById(R.id.board);
        left = (Button) findViewById(R.id.Left);
        right = (Button) findViewById(R.id.Right);
        newGame = (Button) findViewById(R.id.NewGame);
        tv = (TextView) findViewById(R.id.textView1);

        tv.setText("score = " + score);
        c = new Cell();
    }
}

```

```

getRandEq();
row = 0;
col = random.nextInt(5);
;
r2 = row;
c2 = col;

t = new GameThread();//create game thread
t.start();

right.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        /**
         * this button is used to move the card to the right
         *
         * can go to the right
         *
         * **/
        while it
        {
            if (col < 4) {
                if (boardview.checkEmpty(row, col, r2, col + 1))

                    c2 = col + 1;
                    if (boardview.check(row, col, r2, c2)) {
                        row = r2;
                        col = c2;
                    }
            }
        }
    }
});

left.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        /**
         * this button is used to move the card to the left
         *
         * go to the left
         *
         * **/
        while it can
        {
            if (col > 0) {
                if (boardview.checkEmpty(row, col, r2, col - 1))

                    c2 = col - 1;
                    if (boardview.check(row, col, r2, c2)) {
                        row = r2;
                        col = c2;
                    }
            }
        }
    }
});

newGame.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

```

was

```
/**
 * this button is used to start new game it initialise
 * everything and if the game was lost so the thread
 * returned then will create new thread
 *
 * **/
boardview.isLost = false;
boardview.init();
score = 0;
boardview.paint.setColor(Color.BLACK);
tv.invalidate();
boardview.invalidate();

c = new Cell();
getRandEq();
row = 0;
col = random.nextInt(5);
r2 = row;
c2 = col;

if (threadnull) {
    t = new GameThread();
    t.start();
}

try {
    Thread.sleep(150);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

});
}

/**
 * Generate the equation randomly, the equation contain only 2 values and
 * there is only 2 operation addition and subtraction, each value is from
 * the range [0 to 9] the results will be in the range of [-18 to 18]
 *
 * after generating the equation randomly calculate the result based on the
 * operation the result of the equation is calculated
 *
 * **/
public void getRandEq() {
    char op = ' ';
    int res = 0;

    int x = random.nextInt(10);
    int y = random.nextInt(10);

    if (random.nextInt(2) == 1) {
        op = '+';
        res = x + y;
    } else {
        op = '-';
        res = x - y;
    }
}
```



```

        c.s = x + " " + op + " " + y;
        c.result = res;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    class GameThread extends Thread implements Runnable {
        @Override
        public void run() {
            while (true) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        boardview.invalidate();
                        tv.invalidate();
                    }
                });
                boardview.setPosition(c, row, col);
                if (boardview.check(row, col, r2, c2)) {
                    row = r2;
                    col = c2;
                }
                try {
                    sleep(500); // this is the speed of moving the
card to the
                                // bottom for 1 row
                    if (!boardview.checkFlip(row, col, r2 + 1, c2) &&
r2 < 9) {
                        /**
                        * check if we should flip the card or the
card should
                        * be move down to the next row
                        * */
                        r2 = row + 1;
                    } else {
                        /**
                        * here we will flip the card so will
change the

```

will create new

we will increase

removed in the

thread and the

```

        * equation with its result and then we
        * card to play with
        *
        * **/
        c.s = c.result + "";
        boardview.setPosition(c, row, col);

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                boardview.invalidate();
                tv.invalidate();
            }
        });

        if (boardview.checkResults(row, col)) {
            /**
             * if the two results are the same
             *
             * the score and the cards will be
             *
             * checkResults method
             *
             * **/
            score++;
        }
        /**
         * create new cell with new equation
         *
         * **/
        c = new Cell();
        getRandEq();
        row = 0;
        col = random.nextInt(5);
        r2 = row;
        c2 = col;
        if (boardview.checkGameEnd(row, col)) {
            /**
             * check if the game end to stop the
             *
             * game
             *
             * **/
            boardview.islost = true;
            threadnull = true;
            return;
        }
        tv.setText("score = " + score);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}
}
}
```

Flying Flashcards



New Game

score = 0

1 + 6

4

2

-5

<

>

Flying Flashcards



New Game

score = 0

$$0 + 2$$

4

2

7

-5

6

<

>



Saving screenshot...

Flying Flashcards



New Game

score = 1

6 - 4

4

7

-5

6

<

>



Saving screenshot...

Flying Flashcards



New Game

score = 1

6 - 4

4

7

-5

6

<

>

Flying Flashcards



New Game

score = 1

6 - 4

-6

4

2


7

-5

6

<

>

 Saving screenshot...

Flying Flashcards



New Game

score = 2

5 - 5

4

Copied to clipboard.

-6

6

<

>

Flying Flashcards



New Game

score = 2

3 - 3

9

-6

4

0

7

-5

6

<

>



16°



67%



21:19

Flying Flashcards



New Game

score = 2

3 - 3

9

-6

4

0


7

-5

6

<

>

 Saving screenshot...

Flying Flashcards




New Game

score = 2

9	3 - 3		-6	
4	0	7	-5	6

<

>

 Saving screenshot...

Flying Flashcards



New Game

score = 3

2 + 4

9

4

-6

7

-5

6

<

>

Flying Flashcards



New Game

score = 4

$$7 + 0$$

9

-6

4

-1

7

-5

<

>



16°



67%



21:20

Flying Flashcards



New Game

score = 4

7 + 0

9

-6

4


-1

7

-5

<

>

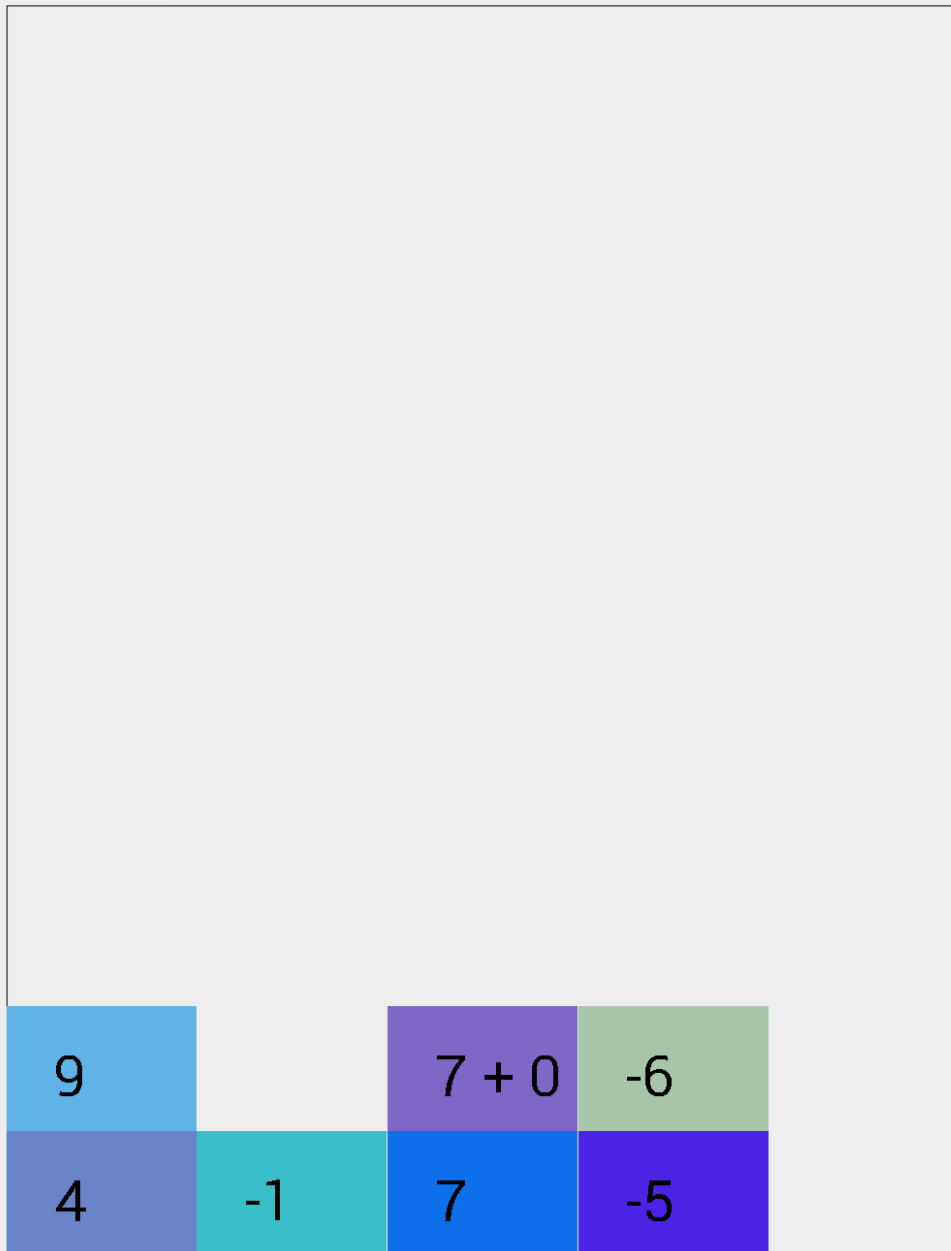
 Saving screenshot...

Flying Flashcards



New Game

score = 4



<

>



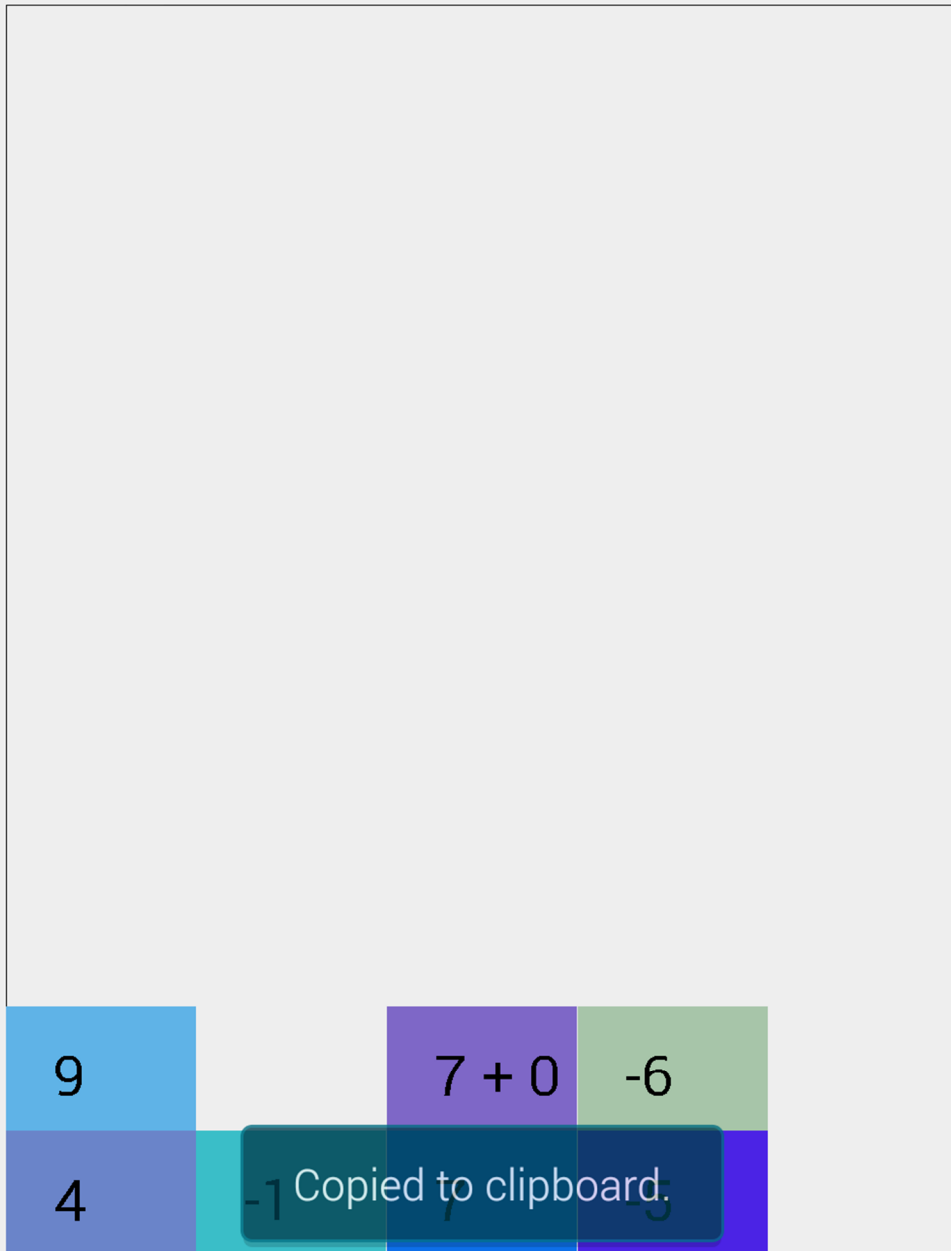
Saving screenshot...

Flying Flashcards



New Game

score = 4



<

>

Flying Flashcards



New Game

score = 5

7 + 8

9

-6

4

-1

13

-5

6

<

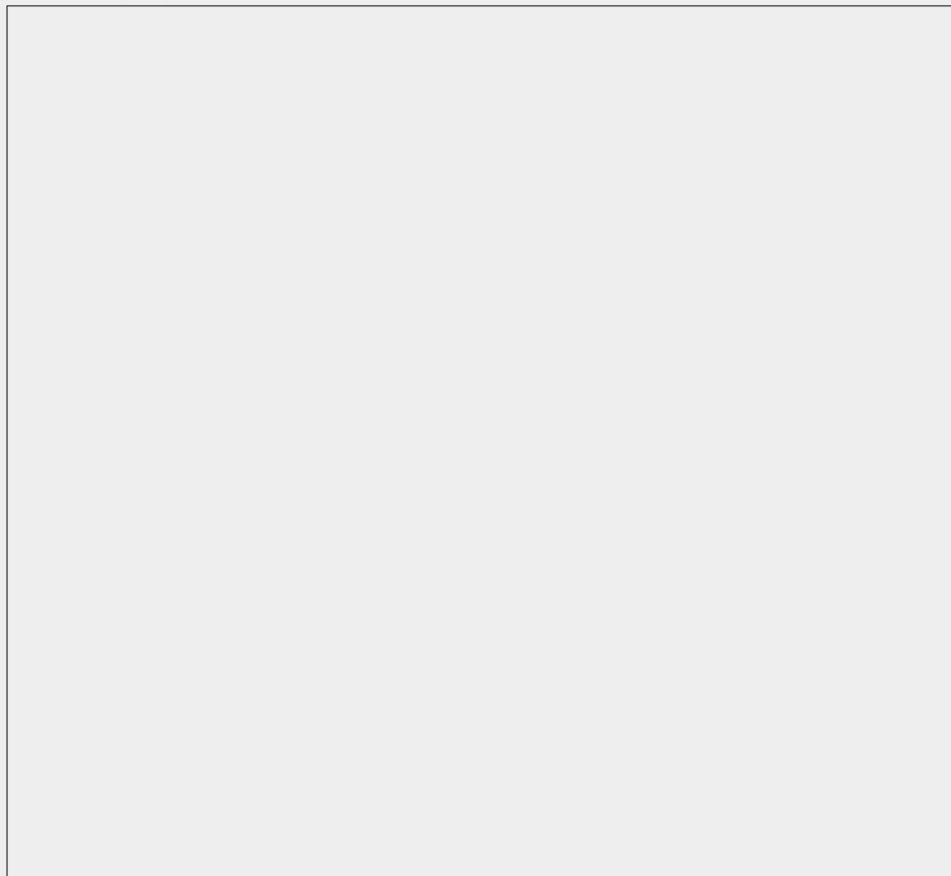
>

Flying Flashcards



New Game

score = 5



9 + 9			-9	-1
9	15	8	-6	5
4	-1	13	-5	6

<

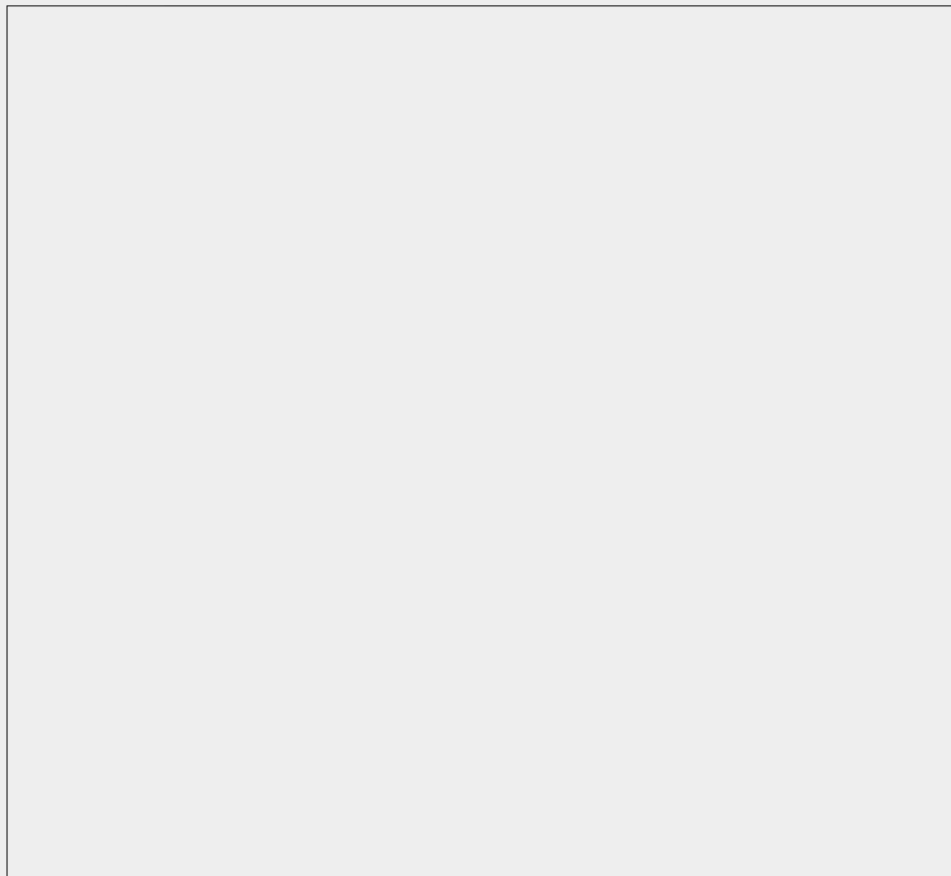
>

Flying Flashcards



New Game

score = 5



18	-4	7 - 7	-9	-1
9	15	8	-6	5
4	-1	13	-5	6

<

>

Flying Flashcards



New Game

score = 5

	9 + 4			
3	14			3
7	-6	1		4
5	12	-1	14	11
9	11	1	-2	13
1	15	-5	4	7
3	4	5	9	10
18	-4	0	-9	-1
9	15	8	-6	5
4	-1	13	-5	6

<

>

Flying Flashcards

New Game

score = 5

	13			-6
3	14			3
7	-6	1	6	4
5	12	-1	14	11
9	11	1	2	13
1	15	-5	4	7
3	4	5	9	10
18	-4	0	-9	-1
9	15	8	-6	5
4	-1	13	-5	6

<

>

Flying Flashcards



New Game

score = 5

	13			-6
3	14			3
7	-6	1	6	4
5	12	-1	14	11
9	11	1	2	13
1	15	-5	4	7
3	4	5	9	10
18	-4	0	-9	-1
9	15	8	-6	5
4	-1	13	-5	6

<

>

You lose