

Processing (Java): Bolle

Introduzione

Il gioco consiste nel calcolare velocemente la somma di due numeri che scendono dall'alto e scrivere la tale somma. I numeri nel gioco originale (TuxMath) erano racchiusi in una bolla che scoppia se tocca terra. Nel tutorial ci si limita a far scendere i due numeri senza particolari effetti grafici.

The game to code

Two bubbles appear on the top of the screen (left hand side) with two numbers inside them. They start falling down. There appear two rectangular areas on the right hand side of the screen. In the lower one the user is asked to enter the sum of the two numbers. The user's answer is seen right under the sentence that is asking for the sum. When the user presses enter/return in the text box above the text indicates whether the answer is correct, repeating the correct answer right below, or wrong, giving the right answer below. There is a counter for correct answers and total number of answers given by the user until the game is over. After each answer new bubbles will appear on the screen with two new numbers on them. The game is over when the user does not manage to give an answer and the bubbles reach the bottom of the screen. Then in the lower rectangular area a text appears saying that the game is over and giving the number of correct answers and the total number of attempts in the game.

You should write the code iteratively, i.e first create and initialize the basic variables, write the `setup` function to create the screen and the `draw` function that implements the animation. Afterwards change the code where necessary, add one or two other functions in order not to repeat code, and get the final game. Please try to code it yourself first, without looking at the given example code below!

One possible solution in Processing

Since this game can be coded in many different ways we give here one basic solution. Afterwards using the suggestions in the "Suggestions for improvements" section, you can change the program in the way you want.

Oggetti utilizzati

The objects used in the game are:

- two bubbles (two circles) with numbers written in their center that appear in the top of the screen and start falling down
- two rectangular areas where text is to be written dynamically during the game

Variabili

We choose using global variables and write functions that do not need to pass variables as input/output. The variables that need to be created are:

- variables that define `x` and `y` coordinates of the two bubbles
- the two numbers that should appear inside these bubbles, and their sum
- the variable for storing the answer given by the user
- a state variable to keep track of the state of the game (new game, correct answer, wrong answer, game over) which is used for deciding on the text to appear in the rectangular areas and for the initialization of numbers to appear in the bubbles
- two counters, one for correct answers, one for total answers given by the user
- variables for keeping the messages given in the two text areas

```
// variables for positioning the bubbles
```

```

float circleY = 25;
float circleX1 = 50;
float circleX2 = 150;

// the values in the bubbles and their sum
int num1 = int(random(100));
int num2 = int(random(100));
int realSum = num1 + num2;

// for keeping track of the answers given by the user
int count = 0;    // count of correct answers
int total = 0;    // total additions in the game

String solution = ""; // used to store user input - the sum of the two numbers - in type String
int numSolution;      // integer version of the user input

int state = 0;        // state of the game:
                      // 0: a new set of numbers appear in the bubbles
                      // 1: correct sum is given by the user
                      // 2: wrong sum is given by the user
                      // 3: bubbles have reached the bottom of the screen without an answer from the user

String message1 = "", message2 = ""; // messages to be displayed to the user
                                     // message1: when state = 0 or 1, appears in the lower text box
                                     // message2: when state = 2 or 3, appears in the upper text box

```

The setup function

In this basic version of the game `setup` function is used only to create the screen for the game.

```

void setup(){
    size(600, 400);
}

```

The draw function

Assumptions: The two bubbles are created at the same y coordinate, at the same time and they fall down in parallel at the same speed.

1. Start with your color choice for: the background, the bubbles and the two text areas. You can use fixed colors or also random colors.
2. Create the bubbles and write the numbers inside the bubbles.
3. To get the effect of bubbles falling what are you supposed to do? Write the code that animates the bubbles.
4. How do you check whether the bubbles have reached the bottom of the screen? Check whether they have reached bottom. What happens when/if they reach the bottom?

```

void draw(){
    background(200);    // grey

    if (state == 1 || state == 2) { // a right or wrong answer has been given by the user
        delay(2000);
        state = 0;
    }
    fill(255, 10, 10); // green; can be changed with any color
    rect(300, 250, 300, 30);
}

```

```

myTextBox();

fill(0, 255, 0); // green
ellipse(circleX1, circleY, 20, 20);
fill(0, 0, 0); // black
text(num1, circleX1-7, circleY+5);
fill(0, 255, 0);
ellipse(circleX2, circleY, 20, 20);
fill(0, 0, 0);
text(num2, circleX2-7, circleY+5);
circleY = circleY + 1;
if (circleY > height && (key != RETURN || key != ENTER)) { // bubbles arrive at the bottom
    state = 3;
    myTextBox();
    noLoop(); // stop the draw loop
}
}

```

Notice the `delay()` function in the above code? It is necessary to stop the animation for a period of time so that the messages in the text fields are seen. In order for the `delay()` function to work correctly we need to add one line at the head of our program which will include a library.

```
import processing.serial.*; // we need this to put some delays in the game
```

keyPressed function

Assumptions: we assume that the user presses only numeric keys, and **enter** or **return**. This assumption can be canceled in another version of the code where the keys pressed could be checked to be numeric and a warning issued if the user presses any other keys.

We should keep track of the keys pressed by the user. This function is also used for deciding whether the answer of the user is correct or wrong.

Write a function which:

- checks whether the key pressed is **enter** or **return** (meaning that (s)he has finished writing her answer)
- if *yes*, then we should check whether the answer is correct.
- if *yes*, then we should issue a message saying that the answer is correct, do some bookkeeping (update number of correct answers and total attempts), and restart the game with new numbers
- if *no*, then we should issue a message saying that the answer is wrong, give the correct answer, do some bookkeeping (increase the number of total attempts) and restart the game with new numbers
- if *no* (and the key pressed is numeric), we add the new key character to the string of other keys pressed before.

NB: remember that the sum of two numbers is an integer however the keys pressed are characters. Do the necessary type conversion in order to be able to compare the answer given by the user to the real sum of the two numbers.

```

void keyPressed()
{
    if (key == RETURN || key == ENTER) { // user input is finished; check if the result is correct
        numSolution = int(solution); // get the integer value of the user input
        total++; // increase the number of total answers given by the user by 1
        if (numSolution == realSum) { // correct answer
            state = 1;
            count++; // increase the number of correct answers by 1
            myTextBox();
        }
    }
}

```

```

        // restart the game with new numbers
        num1 = int(random(100));
        num2 = int(random(100));
        realSum = num1 + num2;
        solution = "";           // initialize the variable for keeping keys pressed to empty
        circleY = 25;
        return;
    } else { // wrong answer
        state = 2;
        myTextBox();
        // restart the game with new numbers
        num1 = int(random(100));
        num2 = int(random(100));
        realSum = num1 + num2;
        solution = "";
        circleY = 25;
        return;
    }
}
else if (key >= '0' && key <= '9') // user continues writing
{
    solution = solution + key;      // add the latest key pressed to the string of keys pressed
}
}

```

myTextBox function

To have a more readable and compact code we write a function that will print the right text in the right text box for each state of the game, i.e.:

- ask for the sum (in the lower text box) when a new pair of numbers appear in the bubbles
- print a message (in the upper text box) when a correct answer is given
- print a message (in the upper text box) when a wrong answer is given
- print a message (in the lower text box) when game is over

```

void myTextBox() { // to communicate with the user

    switch(state) {
        case 0: // game start; a new set of numbers
            fill(255); // white; can be changed with any color
            rect(300, 350, 300, 30);
            fill(0);
            message1 = "What is the sum of these numbers?\n";
            text(message1+solution, 310, 365);
            break;
        case 1: // correct solution
            fill(255, 20, 20); // red; can be changed with any color
            rect(300, 250, 300, 30);
            fill(0);
            message2 = "Correct! \n ";
            text(message2+solution, 310, 265);
            break;
        case 2: // wrong solution
            fill(255, 20, 20); // red
            rect(300, 250, 300, 30);

```

```

        fill(0);
        message2 = "Wrong answer! It was \n";
        text(message2+realSum, 310, 265);
        break;
    case 3: // bubbles out of screen, no answer, game over
        fill(255); // white
        rect(300, 350, 300, 30);
        fill(0);
        message1 = "Time out! Game over with "+count+"\ncorrect answers out of "+total;
        text(message1, 310, 365);
    }
}

```

Suggestions for improvements

- Add an explosion effect to the bubbles when they reach the bottom of the screen
- Add a timer to the game and put a timeout. The game ends either when the timeout is reached or when the bubbles reach the end of the screen.
- Add a new level when a certain count of correct answers is reached. In this new level you can add a new bubble that carries a random arithmetic operator + or - which will be the operation to do between the two numbers. Or you can have a third number in the new bubble and do the sum of three numbers instead.
- Any other improvements you can think of??
- Any graphical/animation improvements?

Just go for it!