Classes: Arrays

1. Learning aims

At the end of these exercises you should be able to:

- understand why classes are useful
- instantiate a class and use its functionality
- save pointers to the objects in an array
- make simple applications with arrays objects
- make animations with images stored in an array

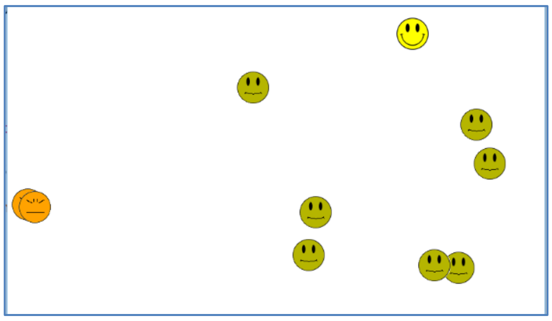
2. Course Material

- Processing : A programming Handbook for Visual Designers and Artists (C. Reas & B. Fry) – Second Edition
    - Chapter **Objects** (p. 359 > p. 376)
- https://www.youtube.com/results?search_query=processing+tutorial+from+beginner+to+games

3. Lab exercises

3.1. Sketch Demo_03_SmileyWorld

In this application 10 smileys are stored in an array and move around. Speed can be increased or decreased. Different states are possible (bouncing/highest).



❖ **Smiley class**:

Data members:

xPos, yPos, diameter (float)

speed, xDirection, yDirection (float)

imgSmiley (PImage)

isHighest, isSleeping (boolean)

Constructor:

Smiley(float xPos, float yPos, PImage imgSmiley)

All the data members are initialized. The speed is a random value between 60 and 180. The booleans isHighest and isSleeping are initialized to false.

Methods:

move()

adjust the position variables and make the smiley bouncing the borders. You will notice that the smileys go very fast. Each frame the position changes, so multiply the position with 1/framerate, so he changes it only once in 1 second!

display()

Draws the part of the bitmap according to its state:

- If the smiley is at the highest position (isHighest is true), then the happy smiley is drawn.
- If the smiley is near the screen boundaries (20 pixels border), the scared smiley is drawn.
  If the smiley is sleeping, the sleeping smiley is drawn
- If none of the above conditions is met, the neutral smiley should be drawn.

increaseSpeed()
Multiplies the speed with the value 1.05.

decreaseSpeed()
Divides the speed by the value 1.05.

hitTest (float mx, float my)

If the arguments (mx and my) are within the boundaries of the surrounding rectangle then the sleeping state of the smiley changes (TIP: cf last week "lights" exercise!).

Getter: float getYPosition()

this function returns the Y-position of a smiley object

Setter: void setHighest(Boolean highest)

this method sets the data member "isHighest" of the object for being the highest to true.

❖ Main application

setup():

Create the smiley class array and assign 10 Smiley objects to the array, filled with random values for the position and the image diameter passed as arguments to the constructor. Size of the sketch is 842 x 480.

draw():

- Move each smiley by calling their move-method
- set the highest smiley
- display each smiley by calling their display-method with a correct image

keyPressed():

- increase the speed of all smileys when the UP-key is pressed
- decrease the speed of all smileys when the DOWN-key is pressed

mousePressed():

- call the **hitTest** method passing the mouse position as arguments when the left mouse is pressed

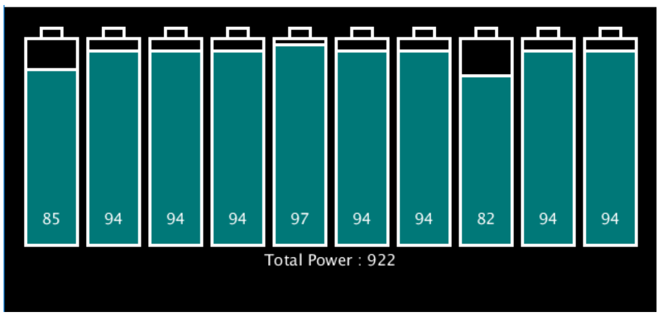
Smiley GetHighestSmiley():

The awake smiley at the highest position should show the happy smiley. Make a function that determines which awake smiley is at the highest position (use getYPosition()). The one with the smallest y-value is situated at the highest position.

Call this function **in the draw** and notify the resulting smiley (using setHighest) that it is at the highest position. Mind that the previous highest smileys are no longer at the highest position, thus also change the data member *isHighest* for these ones (TIP: looping over the array and setting the data member to false, using the setter-method).

3.2. Sketch Lab_03_01_PowerBank

You will make a power bank, composed of 10 batteries. Each second the power of the batteries is reduced and when pressing the "r" key, the batteries are filled.



❖ **Battery class**:

Data members:

- the position of the left, top corner (X and Y)
- the width and the height (= 200px) of the battery
- the current power (starts at 100 and reduces to 0)
- the speed with which the current power reduces (random value between 1 and 5 (included))

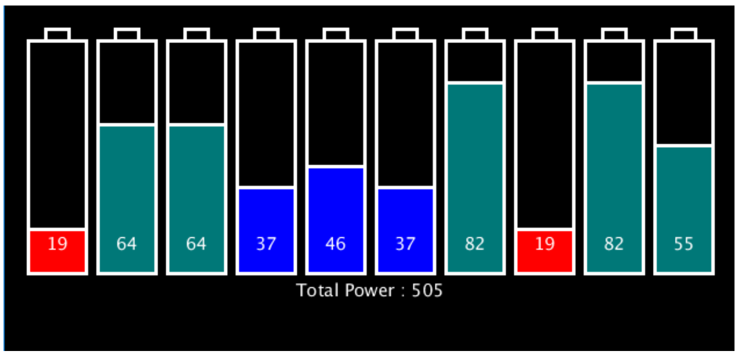
Constructor:

Make an appropriate **constructor**. The width of the battery is always 1/5 of the height of the battery.

Methods:

- **display()**: in which the battery is displayed (cf printscreen). The current power is displayed on the battery. The colour of the background varies:
    - current power > 50 : RGB (0, 120, 120)
    - current power between 50 and 26: RGB(0,0,255)
    - current power between 25 and 0: RGB(255,0,0)

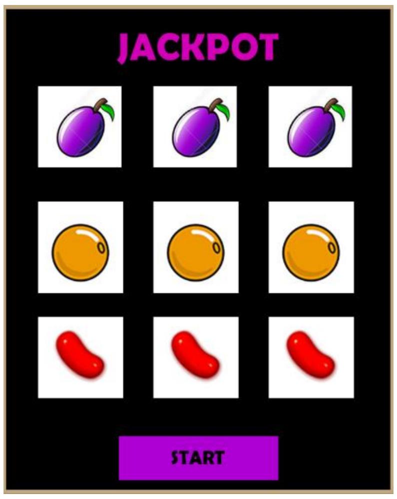
- **reducePower()**: in which the current power of the battery is reduced by the speed. Make sure the minimum power is 0, and cannot become negative.
- **raisePower()**: in which the current power of the battery is raised by the speed. Make sure the maximum power is 100.



❖ Main Application:

- The size of the window is 520 x 300 and the background is black.
- The power bank is composed of an array with 10 batteries. Make an array in which 10 batteries are stored and display the batteries. Make use of a for-loop.
- Each second the power of the batteries is reduced by their appropriate speed.
- When the user presses the r-key, the power is raised of all the batteries once with the speed value.
- When the user presses the right mouse-button, the application stops. Clicking the left mouse-button, the batteries start reducing again from where it stopped.
- Count the total power of the batteries and display this value under the batteries.

3.3. Sketch Lab_03_04_Jackpot



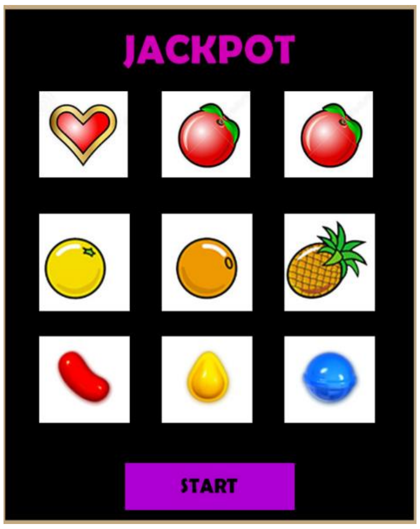
The JackPot application consists of 3 games, a game with icons, fruits and candy.

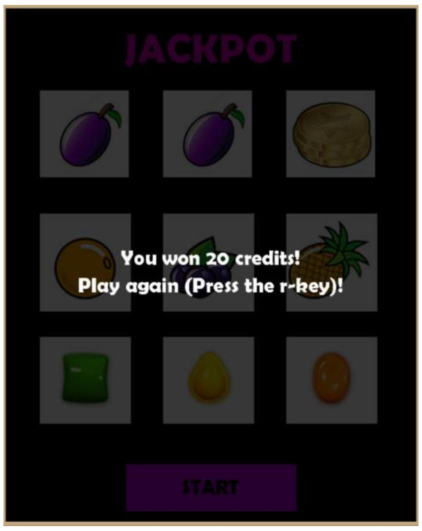
By clicking the start-button, the images are changing for 2 seconds. When the jackpot stops, the images are evaluated.

When 2 identical images are generated in 1 game, the player earns 20 credits. When 3 identical images are generated, the player earns 100 credits per game.
So if you have 3 identical per game, you win 300 credits!

Details:

- The size of the window is 600 x 750 with a black background;
- The title is JACKPOT: color (211, 0, 184) and in Font BerlinSansFBDemi-Bold-60
- Create a **class** "**Jackpot**". In the constructor (***int xPos, int yPos, PImage img, int numElements***), fill an array (depending on numElements) and store the correct images using the image (**No photoshop, use the get() method!).**
- Create 3 objects in the main application. The first row is positioned at (50,120) and display the images.
- Random index numbers (of array) for the 3 games are generated during 2 seconds and the corresponding images are displayed, the "START" button is not shown while generating.
- Visualize the appropriate message after evaluating the generated images.
- Create a **method** to **generate** the **numbers** for each game. Make use of an array (numbers[3]) in which 3 random values(int) are stored.
- Create a **display method**, that selects the 3 images of the image set ,initialized in that specific object, with the correct indexes that are saved in the numbers array.

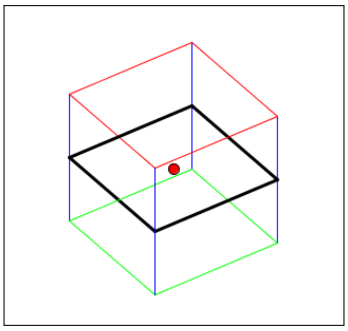




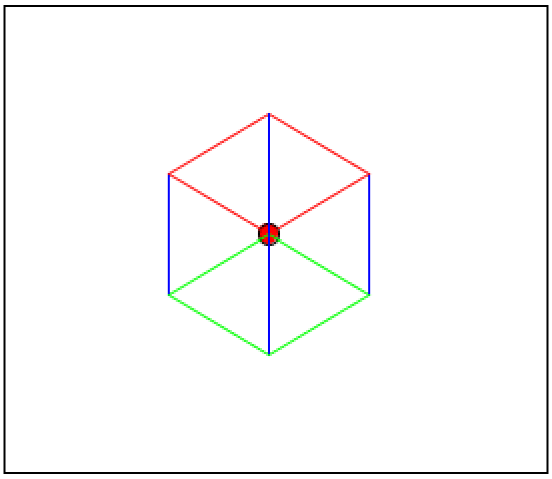
You won 20 credits!
Play again (Press the r-key)!

3.4. Sketch Lab_03_05_CubeControl

A 3D cube can be represented by a base plane (black) that is shifted up (red top plane) and down (green bottom plane).

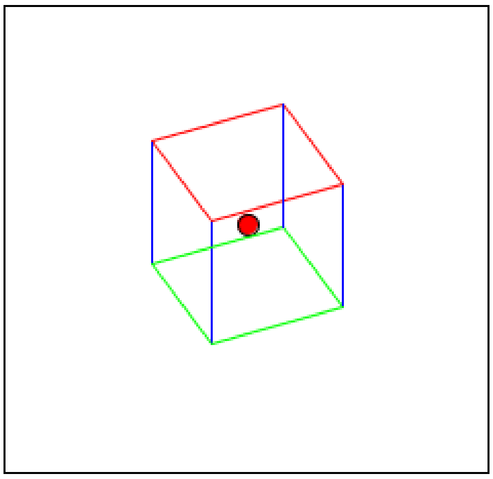


Create an application with rotating cubes.



-

basic Cube with center point (rotation = 0)

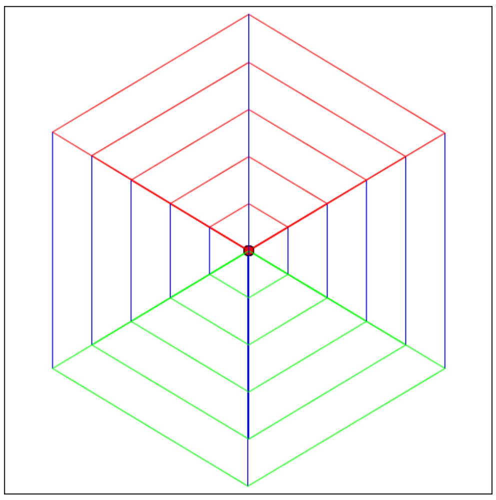


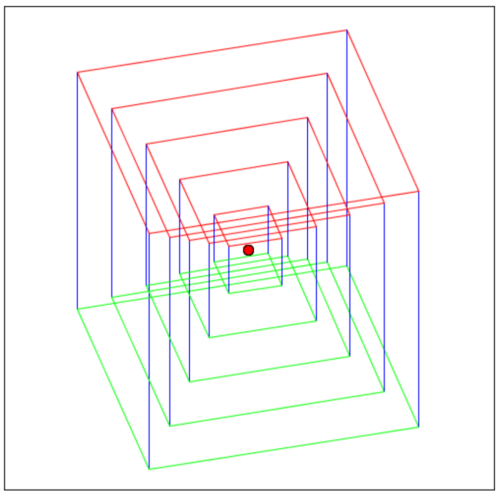
Cube rotated 20 degrees

❖ The window is 500 x 500 with a white background

❖ ❖ Create a class **Cube** to visualize 1 Cube. The Cube has a center position (x and y), a size in the X direction, a size in the Y direction, a color, a rotation angle and 8 vertices.

- o Create data members for the **center position** (x and y) and the **size** (vertical and horizontal) of the Cube, the **color** and **2 arrays** to store the X and Y positions of the 4 points of the base plane. The initial rotation angle is zero.
- o Create a constructor to specify the center position and size (vertical and horizontal) of the cube. The vertical measure is 0.6 * the horizontal size.
- o Create a method calculate () to calculate the x and y positions of the 4 vertices of the black base plane and store them in the arrays. The X positions can be calculated with cos(radians(rotation angle + i * 90))) * sizeX). The Y positions are calculated in a similar way.
- o Make a method rotate() to rotate the cube. In this method the rotation angle increases by 10 degrees each time.
- o A drawCube() method to draw the top and bottom plane in red and green. The vertical lines are blue.
- o Create an array with Cube-objects in the main program. The array contains 5 cubes.
- o When the left mouse button is pressed, each cube rotates.





4. Reflection

What is the difference between a class and an instance of a class?
What is the use of data members in a class?
When is the constructor of a class called?
What is constructor overloading?
What is the advantage of storing pointers in array?

5. Saving instructions

- Make a folder named 1DAE***xx***_PROGFA**2_03**_***name_firstname***
  (e.g. 1DAE03_PROGFA**2_03**_Van_der_Veken_Jan)

- Add all the folders with the corresponding Lab03 pde-files.