Project follow-up and conception choices iteration 4.

For this iteration we had 3 main tasks:

- Guarantee the integrity of the user's projects
- Guarantee the confidentiality of the user's projects
- Handle copy-paste of shapes in the canvas

Hash

We have chosen to add a hash of the (not encrypted) .bin files and to add it in the database. After a successful decryption of a file, the result is hashed and compared to the hash in the database. The hash was necessary on top of the encryption, since an encryption of a modified file can give a binary result that looks legitimate.

In practical terms, SHA-256 is just as secure as SHA-384 or SHA-512. We can't produce collisions in any of them with current or foreseeable technology, so the security you get is identical.

From a non-security perspective, the reasons to choose SHA-256 over the longer digests are more easily apparent: it's smaller, requiring less bandwidth to store and transmit, less memory and in many cases less processing power to compute. (There are cases where SHA-512 is faster and more efficient.)

Third, there are likely compatibility issues. Since virtually no one uses certs with SHA-384 or SHA-512, you're far more likely to run into systems that don't understand them. There are probably fewer issues now than in the past, but again, you're buying yourself risk for no gain.

So, at the present time, there are no clear advantages to choosing SHA-384 or SHA-512, but there are obvious disadvantages. This is why SHA-256 is the universal choice for modern certs for websites. Also, Java 8 has a class that provides inbuilt *MessageDigest* for SHA-256 hashing

At the import, the file is decrypted with the user provided password and the hash is added to the database, otherwise the hash is added or updated when saving a project.

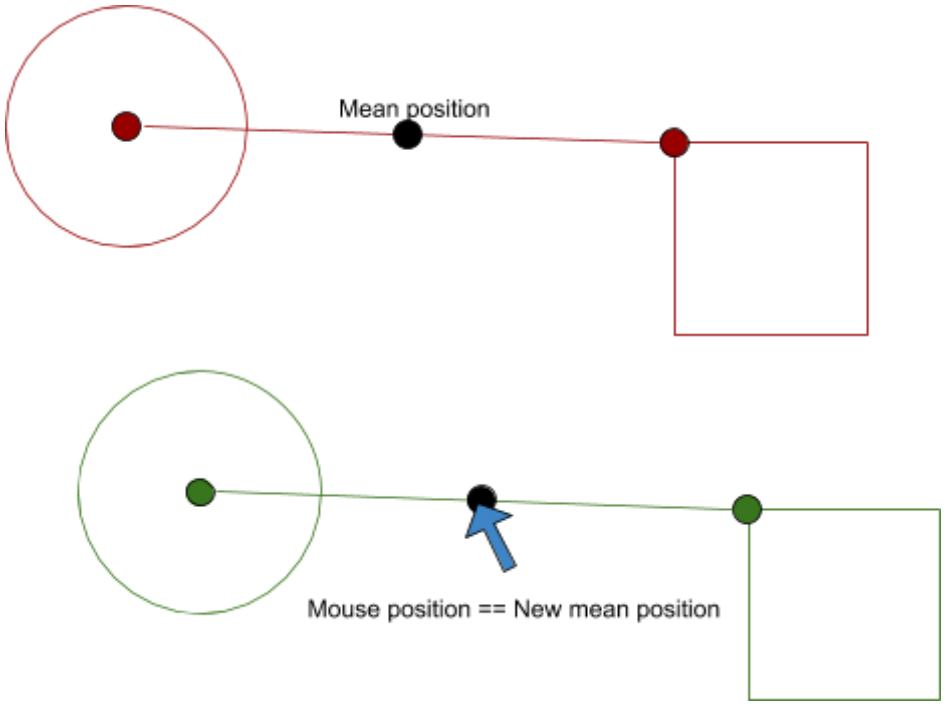
Copy-Paste

Primary Coordinates were added to shapes as an anchor point. The rest of the shape is defined with respects to them. Each Shape was also given a copy-constructor.
The canvas received a clipboard containing the shapes that have been copied. This clipboard is independent from the shapes on the canvas. It is possible, for instance, to copy a Square, delete it and then paste it somewhere else.

Two actions were added to the editor. Namely copy and paste that can be called using the ctrl-c and ctrl-v shortcuts respectively.

- The first action empties the clipboard and then copies the selected shapes from the canvas to the clipboard (only the reference is copied).
- The second action is pasting. It receives the coordinates of the mouse at the moment the ctrl-v shortcut was pressed. It first computes the mean position of all the anchor points of the shapes in the clipboard. This is used as a reference point from which the position of the pasted shapes will be computed. The mean position of the pasted shapes will be at the mouse position received.

  Each Shape to paste is constructed using the copy-constructor on the original Shape and then 'translated' to its target position relative to the mouse. Finally, minor adjustments are made depending on the type of the Shape.



  Example: Let's imagine the canvas contains a circle and a square arranged like the red ones in the image above. That they are selected and then copied to the clipboard. When ctrl-v is pressed with the mouse at the position of the blue arrow on the image, the collection of shapes in the selection follow the same translation as the mouse from the original mean position. Visually resulting in the addition of the green circle and square to the canvas.

Confidentiality

When saving a project, the user can choose and add a password to encrypt the .bin files of the folder of the current project. When encrypted, files have a .enc extension and the content is totally unreadable. When the user wants to open his project again he has to enter the correct password in order to retrieve his project correctly. The files are encrypted with symmetric cryptography, more specifically with AES, because it's much faster than asymmetric cryptography. For generating a password for the secret key, PBKDF2 was used. It's a derivation function that will stretch the key by adding a salt to the password and hash it

multiple times in order to make bruteforce and rainbow tables almost useless. Rainbow tables are unusable thanks to the salt and bruteforce is futile because of the computational work that is needed for the 10 000 iterations that we chose to execute.

In the import, the password is asked to the user and tested by decrypting the project (which also allows for the creation of the hash), then the crypted file is simply copied and renamed in the user folder.