Quality

Lack of quality means a **lower grade**.
The code should be easy-to-read and easy-to-maintain and follow the seen programming rules. So it is entirely up to you that the classes follow the coding rules such as: const correctness, rule of 5, initialization, naming, formatting, releasing resources,...

The project should build without errors and without warnings in both x86 and x64.

There should be no code that doesn't correspond with this assignment (no copy / paste from other exercises).
Names of variables, functions, enumerations, structs,… should be meaningful and follow the agreed rules.

Avoid non-performant code, for example, unnecessary creation of objects each frame is out of the question. Avoid global variables. Prefer local variables above data members.

General

You have to create a puzzle game. The puzzle consists of 4 tiles allowing the composition of 1 of 6 animals. Each tile shows a particular part of the animals and it is possible to browse through the animals of a tile. When all the tiles show parts of the same animal, then the puzzle is solved. Try the given working example.

Technical requirements

It is important that you complete the assignment as requested so that we can verify whether you have acquired some specific programming skills. If your solution works, but if you have solved it in another way, then this results in a lower grade. This concerns for example:

- Defining the Tile class,
- Creating objects of this Tile class,
- Keeping the created Tile objects or their addresses in an array and manipulating these objects via this array.
- Respecting the rule of 5
- …

Given

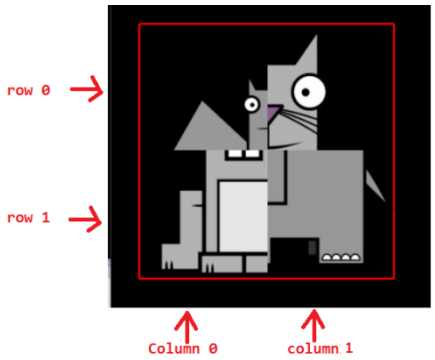
- The folder **Resources** with the necessary images.
- The folder **CodeFiles** containing the Tile.h and Tile.cpp files. The Tile class describes a tile of the puzzle. Only the public interface is given, you decide about the private part. You also determine whether a **destructor** is needed. This public interface is commented, uncomment the method declarations step by step as described further in this document. You also determine yourself which methods can be **const**, this has not been indicated in the given declarations.
- **Ex50.exe** a working example.

Create the project

1. Create a project with name **Prog1DAE<u>xx</u>NameFirstnameEx50** replacing **<u>xx</u>** by your group number and NameFirstname by your name and first name.
2. Perform the necessary steps to use your **09_Framework** files.
3. Copy and add the given class files **Tile.h** and **Tile.cpp**.
4. Copy the given **Resources** folder in the folder of this project.
5. Adapt the **title of the window**, it should mention: Ex50, your name, first name and group.

Assignment

1. Create and draw 4 Tile objects



Each Tile object shows a specific part of the animals. The resources folder contains the images **TilesRC.png** with: **R** indicating the row and **C** the column number of the tile, as indicated on the picture on the left.
For example the image Tiles00.png contains the upper left corner part of the 6 animals.

Tile class

Define the Constructor, the parameters are:

- The rectangular area of the window in which the Tile object should draw its part of an animal.
- The path of the image corresponding to this Tile object.
- The number of animals this image contains.

The constructor chooses randomly which of the animals it will draw. We will call this the **current animal** in the further description.

Define the Draw method. The Tile object draws its part of the current animal in the assigned rectangular area of the window.

Game class

- Create 4 Tile objects, assigning to each Tile object a rectangular area having a width and height equals 128 pixels. The tiles are arranged in a 2x2 grid. Keep the Tile objects or their addresses - you decide about this - in a fixed size array.
- Draw the Tile objects.

2. An activated Tile object draws some extra information

When the location of the mouse is inside the rectangular area of a Tile object, then this Tile object is activated, otherwise it is deactivated.
In case of an activated Tile object some extra drawings are done:

- A semi transparent greyish rectangle is drawn above the animal image.
- Two red rectangular hitregions, one in the bottom left and another one in the bottom right corner. The width and the height of a hit region is $1/4^{th}$ of the width/height of the Tile object's rectangular area.

Tile class

Define the CheckActivation method, when the given position is inside the Tile object its rectangular area, the Tile object gets activated. When it is outside this area then it gets deactivated.

Extend the **Draw** method with the extra drawings in case the Tile object is activated.

Game class

Perform the necessary steps so that a Tile object is activated/deactivated when the mouse is inside/outside its area.

3. Showing the next/previous animal part when clicking a hitregion

Tile class

Define the CheckHit method. When the given position is inside the left/right hitregion of the Tile object, the previous/next animal becomes the current one and *true* is returned. This is a circular operation:

- When the first animal is the current one, clicking on the left hitregion makes the last animal the current one.
- When the last animal is the current one, clicking on the right hitregion makes the first animal the current one.

When the given position is not inside those hitregions, *false* is returned.

Game class

Perform the necessary steps to let the player browse through the animal parts when the left mouse goes down above one of the hitregions.

4. Indicate whether the puzzle is solved

Tile class

Define the **GetCurrentAnimal** method which returns the number (e.g.: 0 to 5) of the current animal.
Define the **Deactivate** method, which deactivates the tile.

Game class

Perform the necessary steps to know whether a puzzle is solved, this is when all the tiles show their part of the same animal.

Draw a border around the puzzle. The border is **red** when the puzzle is **not solved** and **green** when it is **solved**.

Also draw the **name of the animal** when the puzzle is solved. The names are located in the **Names.png** image. The names sequence (from top to bottom) is the same as in the TilesRC.png images (from left to right).

Write performant code, don't check whether the puzzle is solved in each frame, but only when it is really necessary.

When the puzzle is solved, tiles are no longer activated and can't get activated, so the hitregions and greyish rectangle are no longer drawn and clicking on a tile should have no effect.

5. Shuffle the tiles when the r-key goes up

Tile class

Define the **Randomize method** which randomly chooses a current animal.

Game class

When the r-key goes up, call this Randomize function of the Tile objects.

Make sure that the puzzle is not solved after this. So the tiles shouldn’t accidentally all show the same animal after this operation.

As a result one can again solve the puzzle by clicking on the tiles.

6. At start the puzzle should not be solved

After having created the Tile objects, they could accidentally show the same animal. Avoid this by shuffling them - in the same way as in previous step - after creation.

Submission