Quality

Lack of quality means **fewer points.**.

The code must be easy to read and maintain and complies with the programming rules.

The project has to build without errors and without warnings.

There may not be any code that has nothing to do with this assignment (no copy/paste of other exercises).

Names of variables, functions, enumerations, structs,.. must be meaningful and follow the agreed naming rules.

Make sure there are **no memory leaks**, all **dynamic allocated objects** should be **removed** before closing the program.

It is not allowed to use C++ features that have not yet been discussed in the lessons. For example: std::vector, smart pointers…

Online or Onsite

If you are taking this exam online, you are required to be present on the agreed upon online platform (Discord https://discord.gg/HUSaYkXSQD). You will need a webcam and microphone to identify yourself.

Technical requirements

It is important that you perform the assignment **as requested** so that we see if you have **acquired** certain **programming skills**. If your solution works, but if you have solved it in a **different way**, you will get **fewer points**. This includes, for example: The creation and use of the described arrays, classes, enumerations, ...

If you solve this assignment **without** using **classes**, you will **lose** a **considerable** amount of **points**.

Given

A zip file with a working example exe, source file(s) and the resources.

Create the project

1. Create a visual studio project with the framework, using the files in 00_General/FrameworkFunctions_V0.2.4.zip. The name of the project is **Prog1DAExx_Exyy_Familyname_Firstname,** replacing **xx** with your group number, **Familyname** by your name and **Firstname** by your first name, **yy** by the exam number as it can be found in the header of this document.
2. Put your full name and group in the window title.

Assignment

Summary:
This is a game where a chicken tries to cross a busy street.

1. Your name
In Game.h, add your name and group to the windows title bar.

2. Draw the street background
Load the picture street.png and draw it on position 0,0

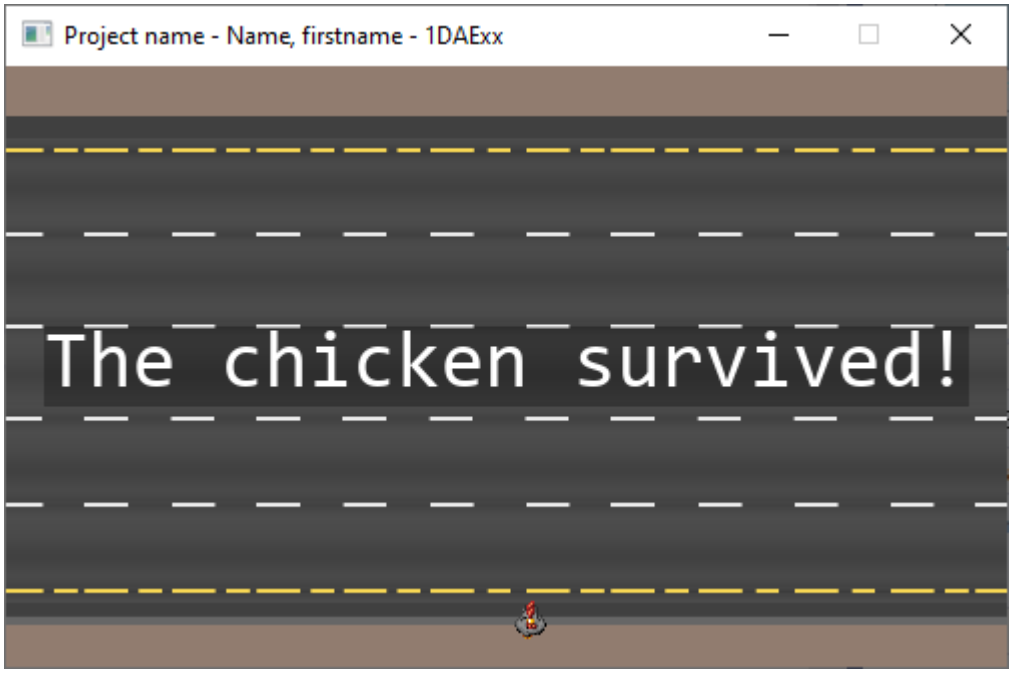
3. The text
All the text that needs to be shown on the window has a size of 40 and uses the provided consola font.

4. Game state
Add and use an enumeration that represents the state of the game. It has the states: play, pause, win, lose, reset. Reset is the initial value of the enumeration variable.

5. The chicken
The chicken is to be implemented in the Game files. It is not a class.
Chicken properties:
- Image is chicken.png.
- Its initial x position is half the width of the window. The initial y position is m_WindowHeight – 20.
- The speed is 100 pixels per second in any direction.

Load the image Chicken.png. It has three frames. For now, only use the first left-most frame. Only at the end of this assignment, we ask you to use the other frames too, as it has a lower priority.
Draw the chicken in such a way that its default (front) frame appears on the pavement, at the top of the window.
When the arrow left or right keys are pressed, the chicken moves in these directions.
When the down arrow key is pressed, the chicken moves downwards, crossing the street.
Pressing down and left or right simultaneously, combines the movement.
The height of the lower pavement is 43 pixels. When the chicken reaches the pavement at the bottom of the street,
- it can no longer move downwards, left, or right,
- a text is drawn “The chicken survived!”, centred on the window, on a darkened rectangle to enhance its readability.

Pressing the ‘r’ key at any moment, resets the chicken to its spawn position at the top pavement and allows it to move again. If any text was being shown, it disappears. The chicken can then cross the street again.
Print key binding and play instructions in the console.



6. The vehicle class:

You can proceed without making the class; but you will lose points.
Create a Vehicle class using the given Vehicle cpp and h files. The public interface cannot be modified. That is the public part of the class definition. You can add private member functions and private member variables.

6.1 Constructor:

Next to the main tasks of a constructor, construct the image path/name string out of the id argument. Load the texture using that string.
The vehicle is not active; it doesn't move when the game starts.
The m_Speed is a random value in the inclusive interval [m_MinSpeed, m_MaxSpeed], where m_MinSpeed is 20 and m_MaxSpeed is 200.
The initial x-position is 0.

6.2 Draw member function:

Draw the texture at the position defined by m_Position.

6.3 SetYPos member function:

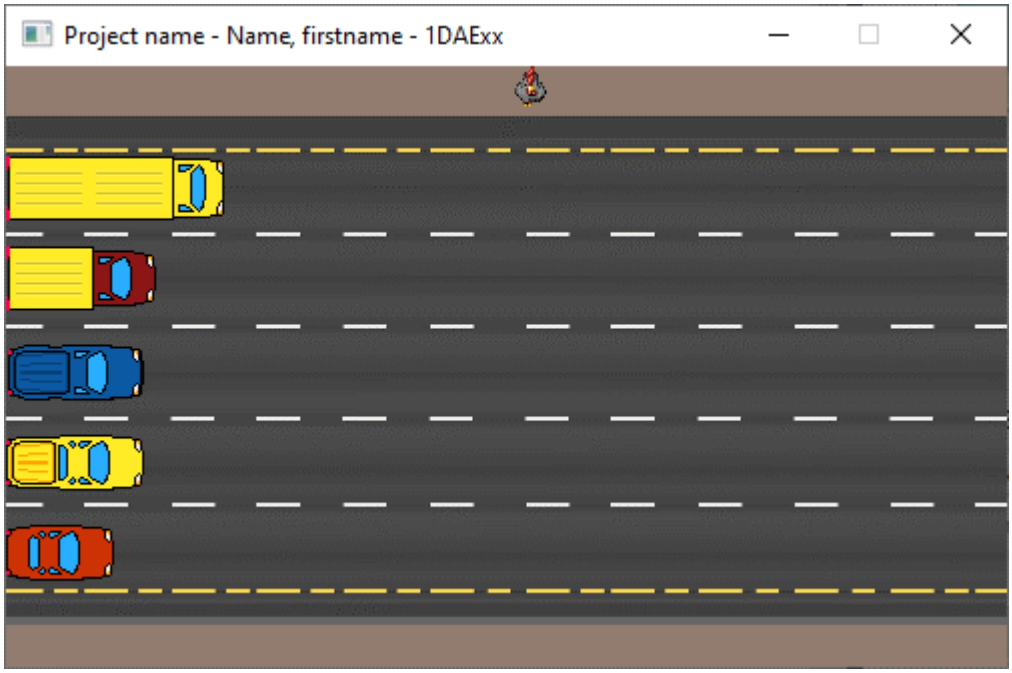
This function updates the m_Position.y value.

7. Add vehicle objects to the Game

Add an array of 5 Vehicle pointers to the Game files and create 5 vehicle objects on the heap while storing each pointer in the array. You can choose not to use an array, but you will lose points. Each Vehicle object should load a different image by passing a number through its constructor. For now, creating the first vehicle passes 1 as parameter to the constructor, creating the 2nd vehicle passes 2, etc…
Update each Vehicle y-position to put the vehicle centred on a lane. The pavement measures 43 pixels and each lane measures 45 pixels. Each vehicle is on different lane. The first vehicle is on the lowest lane, and each next vehicle is on the lane above that as you can see on the screenshot below. Use a loop for this.

8. Draw the vehicles.

They should line up underneath each other. This what the window should look like when running the application:



9. Implement the Reset member function.

In this function:

- Its spawn x-position is randomly chosen left of visible window from 0 till -300.
- Its new speed is randomly chosen between 20 and 200 pixels per second.
- The vehicle is active.

10. Implement the Vehicle Update member function

Perform the necessary steps so that the vehicle moves to the right when it is active. When it leaves the window, it slides back in from the left edge of the window.

11. In Game.cpp: the r-key

When the r-key is pressed, the reset function is called and hey should move from the left to the right.

12. InGame.cpp: Collisions

Add the necessary steps to the Game Update function to have collision detection between the **texture** of the **vehicles** and the **middle** position of the chicken.
When there is a collision:

- the colliding vehicle stops
- the chicken can no longer move
- the other vehicles ignore the collision and continue moving
- a white text appears on a transparent black background “The chicken died”.

13. Game end

When the chicken has successfully crossed the street, a text appears on a transparent black background: “The chicken survived”. All vehicles keep driving.

14. The r-key

Pressing the r-key resets the entire game so that it can be replayed.

15. In Game.cpp: the p-key

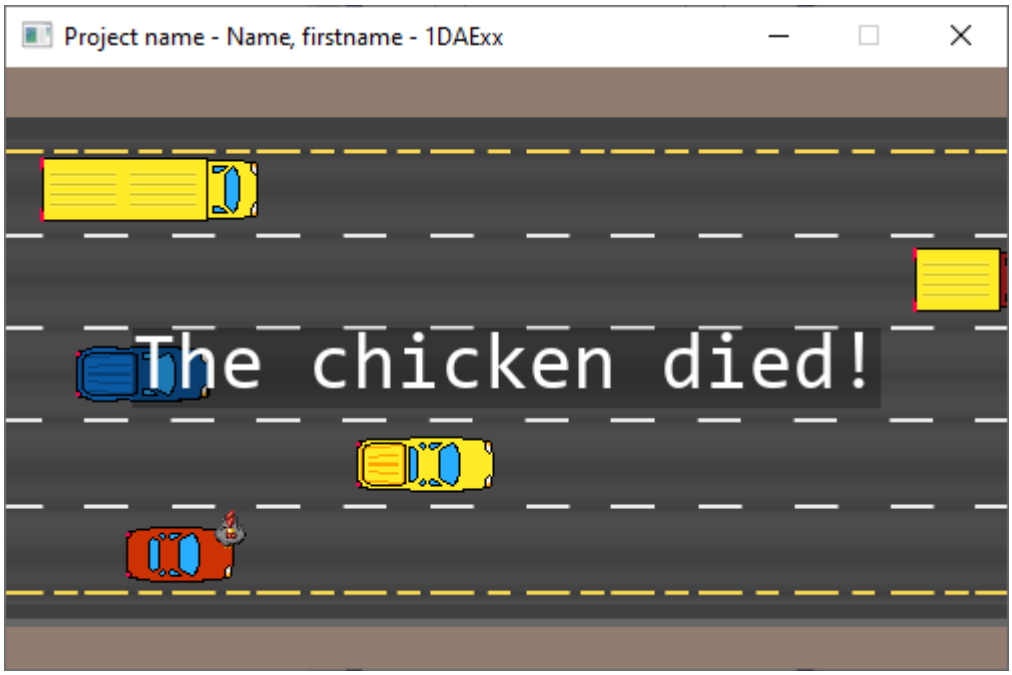
When the p-key is pressed, the game pauses. The chicken can no longer move, nor can the vehicles. A text appears on a transparent black background “Pause”. Press p again to un-pause and remove the text.

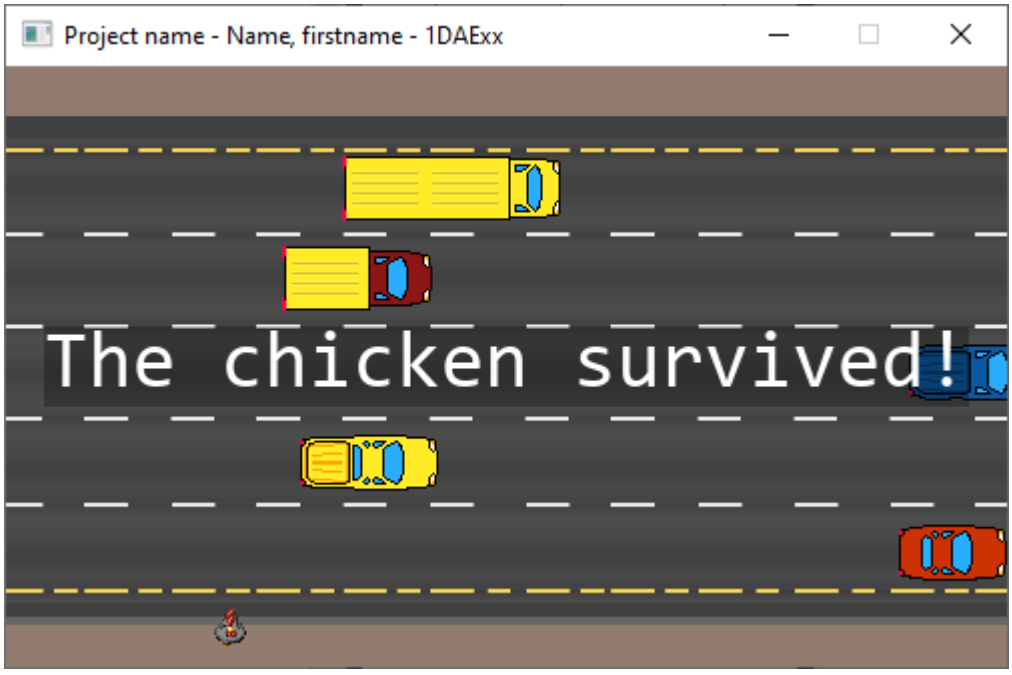
16. Chicken animation

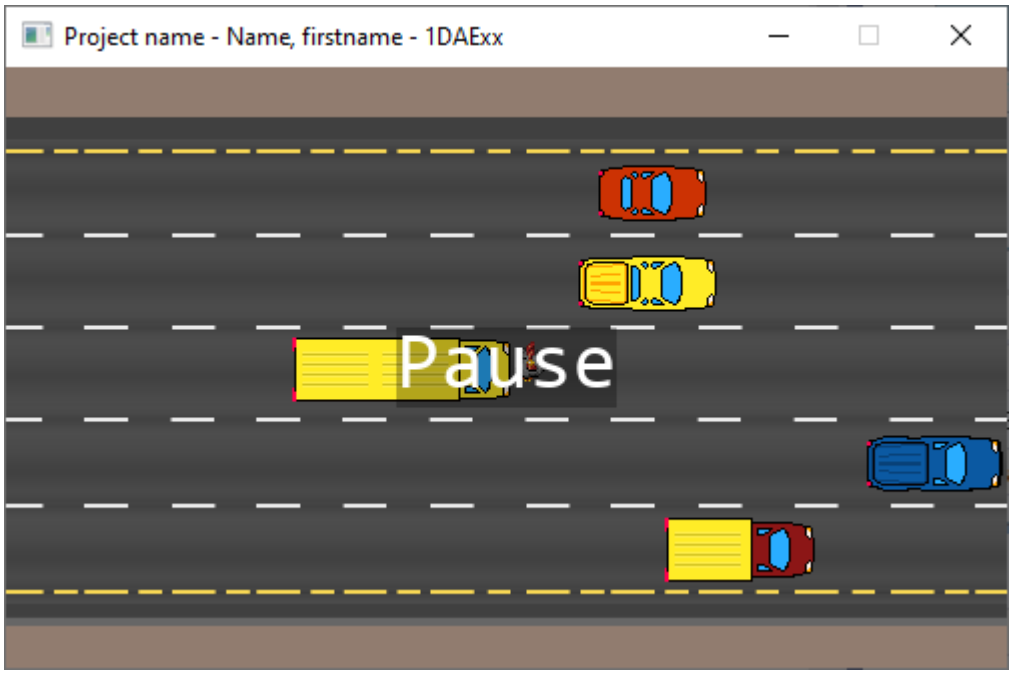
Keep an eye on the exam time. Prevent the already built functionality from breaking by adding this animation.
Each frame of the chicken corresponds to a different state: front, left and right. Use an enumeration to represent that state. Use the enumeration state to decide what frame to draw.

17. Vehicle types

Keep an eye on the exam time. Prevent the already built functionality from breaking by adding this change.
Randomize the order of Vehicle types on the street. Make sure each vehicle type is shown only once. Randomize the number that is used when vehicles are created. Each time the game is started, another order is generated. Resetting does not change the vehicle types.







Screenshot with random vehicle image order, Game is in pause state, a split second before the chicken would be hit.

Submit

Clean the solution. Close Visual Studio. Remove the .vs and the debug and/or the x64 folder.
Check again if your Visual Studio solution folder contains all the necessary code files.
Convert the solution folder to a rar/zip file with the same name as the folder.
Submit this rar / zip file under the leho assignment of this lab exam.
See **for yourself** if this has been done correctly. Incorrect exams – for example, another project, a project with missing code files – get 0 points.