THE DEVELOPMENT OF GRAPHICAL APPLICATION

Lecturer Ilie Borcosi, Student Sandulescu Dragos, „Constantin Brancusi” University of Targu Jiu, Romania

Abstract: In this paper is presented how was realized a graphical application, using the Qt development environment. The Qt development environment is used for the applications development that is based on C++ programming language. It has most of the libraries for graphical projects and also it can use an extra package of own libraries of the designer/user. In Qt Creator application platform can be developed applications for desktop, Android or iOS. It is included a visual debugger and an integrated GUI layout and forms designer. We have used MinGW with the default install and can also use Microsoft Console Debugger when we have compiled source code. For this graphical application created in QT environment are used the object-oriented programming features that makes motion control, graphical collision control and interpreting them in the graphical interface. The graphic interface of the application is represented by a game with planes, where the user controls the aircraft to destroy the enemies, thus obtaining a score that is displayed.

Keywords: graphical application, Qt development environment, object oriented programming

1. Introduction
Creating graphical applications is sometimes a difficult issue. A graphical application can be developed in the Qt environment, using the C++ programming language, adding a set of own libraries, mostly based on elements and graphical functions. Qt Creator is a development environment for many desktop applications, Android applications, and even iOS applications [1]. Qt supports programming in other languages derived from C++: Java, Python, Ruby, C#. A great advantage of this application is that besides the multitude of the C-language libraries and the Qt application, it allows the creation of their own heads, .h extension files that can contain user-defined classes that make programming more readable, thus bringing a great advantage to the programmer [2]. Together with class definitions in the header, the functions can also be defined, making it a very friendly interface. By allowing programming in C language, Qt allows access to object oriented programming, leading to facilitating the creation of classes and objects whose behaviour is defined by the user.

2. Problem formulation
Using object oriented programming, the programmer initially sets a primary class from which all the required classes are derived during code writing. The applications developed in the Qt environmental have a subdirectory containing all of the header defined by the programmer, a subdirectory containing all the source files of the application and a resource subdirectory in which additional application files will be stored (images, sounds, etc.) [3]. The source files are compliant with C language (they have .cpp extension), contain calls of the classes methods, classes builders, operations and instructions for running the application. These features of the Qt environment bring many advantages to the programmer, while making it a very flexible environment.

The graphical application developed is interesting for students, as computer graphics are very attractive, students are motivated to learn.

3. Problem solution
As it was presented, the C language libraries, the Qt environment, as well as classes and libraries (defined by the programmer) were used in the application. From the classes’ point of view, the application contains a main class, the scene class, from which all other classes derive. From the stage class derives a class called vision, which allows the user to view a particular part of the scene, this class being the
basis for deriving all the classes required by the application.

This application consists of a set of Header files (**.h), a set of source files (**.cpp), and a set of multimedia files stored in the Resources subfolder [4].

In the code sequence shown below, the libraries required for game.cpp folder are displayed. Qt's own libraries must be placed between <>, and user-defined libraries must be placed between "" characters. Also, can be included in the source file and the header of the classes and methods previously defined in other header files.

```
scene = new QGraphicsScene();
scene.setSceneRect(0.0, 800, 600);
setBackgroundBrush(QBrush(QImage("t/images/bg.jpg")));
setSceneRect();
setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
setFixedSize(800, 600);
```

Creating the scene is concretized by a new instantiation of the "QGraphicsScene ()" function that assigns pointer scenes the scene characteristics. With the setSceneRect function, the scenes pointer is assigned some features: the coordinates of the top left corner, in this case 0.0, and its dimensions, 800 pixels x 600 pixels. Also in the scene configuration stage, a background image is assigned to it by the setBackgroundBrush function.

A very important aspect to be taken into account is that the scene must contain all the elements related to it, and thus a problem would arise in the running of the program; Objects moving in the scene will generate an increase on the y (vertical) axis because the objects once instantiated are not deleted from the scene alone, this being possible only by some instructions added to the program, namely, checks on the objects in Scene, and if they have exceeded the coordinates of the visual area to be deleted from the scene or memory. Practically without this, the dimensions of the scene will always increase, thus adding unnecessary data to the memory, making the application unnecessary and possibly unstable.

The application provides the user with the ability to control an airplane by controlling his movements in the scene, with the ability to shoot, thus providing a way of defending against enemy airplanes that are created on the screen at a frequency of 2 seconds.

The classes used for the practical application are: Bullet class (that represents bullets), Enemy class (that represents the enemy airplanes), Player class (representing the user-controlled airplane), Score class (that deals with storing a very important aspect of the application because they have to be very well set so that the action does not happen outside the area that is set for viewing (as in the code sequence shown below). The scene class and all other classes derive from the Game class, which basically constitutes the basic class for the other classes [5].

Development of the graphical application, in generally, requires the stages:

a) Determining the application structure.

b) Establish the classes required to develop the application.

c) Determining the functions of each class in order to perform the necessary actions to execute the application. Each class can have its own functions or inherited functions from the base class from which it derives.

d) Adding a graphical interface.

e) Searching for and resolving any errors (bugs).

a) The application will be done graphically, with three sets of visual elements visible in the scene, a set for the enemy airplanes, a set for the user-controlled airplane and a bullet set.

b) To create any Qt application, it must follow steps: creating a scene (using the QGraphicsScene class), creating a scene vision (setting a visual area in the scene using the QGraphicsView class) and create objects in the scene (using the QObject class)

The first step in the development of the application is to establish the stage class, the visual area and its boundaries. The limits of the area that the user of the application visualizes is
The Bullet, Enemy, and Player classes inherit the QObject class functions, thus having the properties of some scene objects. All of the above classes are connected to a timer that has the function of calling certain instructions at a time. In this application, the timer interval is set to 50 milliseconds, so each object will perform a set of instructions at a 50 ms interval. When connected to the same timer, the difference between the object moves will be the number of pixels that each object will advance to a timer call (50ms). Score and Health classes are all user-defined classes, with on-screen display and incremental or decrementing features.

A code sequence are presented below for creating the Player object, reading the events (keystrokes) according to which the airplane must respond by movements, respond to the action button, or shoot [6].

Once the classes, objects, and functions have been defined, the collision conditions have to be added between Bullet class elements and Enemy class elements. This is done with the QList package that generates a list of all the collisions in the scene. A collision is an overlap of coordinates in the scene. In the event of a collision, both the bullet and the enemy must disappear from the scene mainly (not be visible by the player), but for a more optimal program and with fewer resources, they must be deleted both from scene and memory (as below).
d) Application classes are interconnected through the Game class that instantiates the objects of each class. If another class or other function of that class is required to define the functions of a class, the source class, that class, will be included at the beginning of the file. This can also be done with the help of the external function to give the developer the opportunity to use the objects and methods of a class by calling this function. Below are the libraries required for the Enemy.cpp source file, the addresses to other classes and the external function for the Game class.

```cpp
extern Game * game;
```

e) Adding images to the mentioned objects is possible with the QGraphicsPixmapItem library whose function (setPixmap()) can receive as a parameter a saved image in the Resources subdirectory (7).

4. Conclusions
Qt is a development environment based on C++, complementing its functions with a solid
Qt also allows you to create your own Headers (.h extensions) with user-defined functions and classes, thus facilitating the programmer's interaction with the development environment. The graphical application developed is interesting for students, as computer graphics are very attractive, students are motivated to learn.

References