Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

APPOOP

Laboratory work #1

Laboratory description. Game Development based on OOP.

Authors:
Tanașciuc Macarie

Supervisor:
Anastasia Șerșun

Chișinău, 2018

1 Task

Subject: Making a Game with Object-Oriented System Principals **Objectives:** Making a Game with Object-Oriented System Principals

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

1.1 Theory

Object-Oriented programming: Programming using the model, which sees the whole world as the objects and relationship between the objects. According to this programming technique, each object belongs to a class and is derived from this class. Each class has attributes, properties, and methods. For example; Finn is a dog. (In this sentence Finn is a object which is name of the dog and dog is the class) Dogs are age, weight and height. (In this sentence dogs are class also object has many different attributes. Dogs are eat, run and sleep.(In this sentence dogs have many action so those 3 action also are the methods)

2 Implementation

2.1 Encapsulation

The internal structure of an object (its data and its properties) means that it is closed directly from the outside world. In this case, when the value assignment of the object is made, it is protected from misuse. The fact that the internal structure is open to the outside also means that the changes related to the internal structure do not affect the outside world.

Code Implementation:

```
Health::Health(QGraphicsItem *parent): QGraphicsTextItem(parent){
    // initialize the score to 0
    health = 3;

    // draw the text
    setPlainText(QString("Health: ") + QString::number(health)); // Health: 3
    setDefaultTextColor(Qt::red);
    setFont(QFont("times",16));
}

void Health::decrease(){
    health--;
    if (health<=0){ setPlainText(QString("GameOver"));}
    else {setPlainText(QString("Health: ") + QString::number(health));} // Health: 2
}

int Health::getHealth(){
    return health;
}
void Health::setHealth(int hp){
    health = hp;
}
```

Here the following functions:
void Health::decrease();
int Health::getHealth();
void Health::setHealth(int);
Encapsulate the decreasing of health method the method to access what is the current value of the health and the method to set the health in a desired value.

2.2 Abstraction

Abstraction is the process of reducing the information content of a concept. We many things in our minds by abstraction and make sense. For example, a dog; a more general definition is the "live" class. But we never make sentence like "The live is running." Our preference is "Dog is running." Here the dog is isolated from the live.
Health.h header file

```
#ifndef HEALTH_H
#define HEALTH_H

#include <QGraphicsTextItem>

class Health: public QGraphicsTextItem{
public:
    Health(QGraphicsItem * parent=0);
    void decrease();
    int getHealth();
    void setHealth(int);
private:
    int health;
};

#endif // HEALTH_H
```

Access specifiers are the main pillar of implementing abstraction in C++. We can use access specifiers to enforce restrictions on class members. For example:

- Members declared as public in a class, can be accessed from anywhere in the program.
- Members declared as private in a class, can be accessed only from within the class. They are not allowed to be accessed from any part of code outside the class.

2.3 Inheritance

A class in OOP may be inherited from another class. This is also called inheritance. Among the classes, a structure such as an ancestor can be created. Thus, some features do not have to be coded separately for each class, and properties inherit to inherited class. The determinants of access are again determined by which properties are transferred in here.

```
class Player:public QObject, public QGraphicsPixmapItem{
    Q_OBJECT
public:
    Player(QGraphicsItem * parent=0);
    void keyPressEvent(QKeyEvent * event);
public slots:
    void spawn();
};
```

Here the **class Player** function inherits the proprieties of **public QObject** and *public QGraphicsPixmapItem* which means it allows us to use the object proprieties by using base class of objects QObject and the pixmap objects inside class Player.

2.4 Polymorphism

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Real life example of polymorphism, a person at a same time can have different characteristic. Like a man at a same time is a father, a husband, a employee. So a same person posses have different behavior in different situations. This is called polymorphism.

```
Player::Player(QGraphicsItem *parent): QGraphicsPixmapItem(parent){
    // set graphic
    setPixmap(QPixmap(":/images/player.png").scaledToWidth(100, Qt::
    SmoothTransformation));
}

void Player::keyPressEvent(QKeyEvent *event){
    // move the player left and right
    if (event->key() == Qt::Key_Left){
        if (pos().x() > 0)
        setPos(x()-10,y());
    }
    else if (event->key() == Qt::Key_Right){
        if (pos().x() + 100 < 800)
        setPos(x()+10,y());
    }
    // shoot with the spacebar
    else if (event->key() == Qt::Key_Space){
        // create a bullet
        Bullet * bullet = new Bullet();
        bullet->setPos(x()+38,y());
        scene()->addItem(bullet);

    }
}
```

3 Game pics

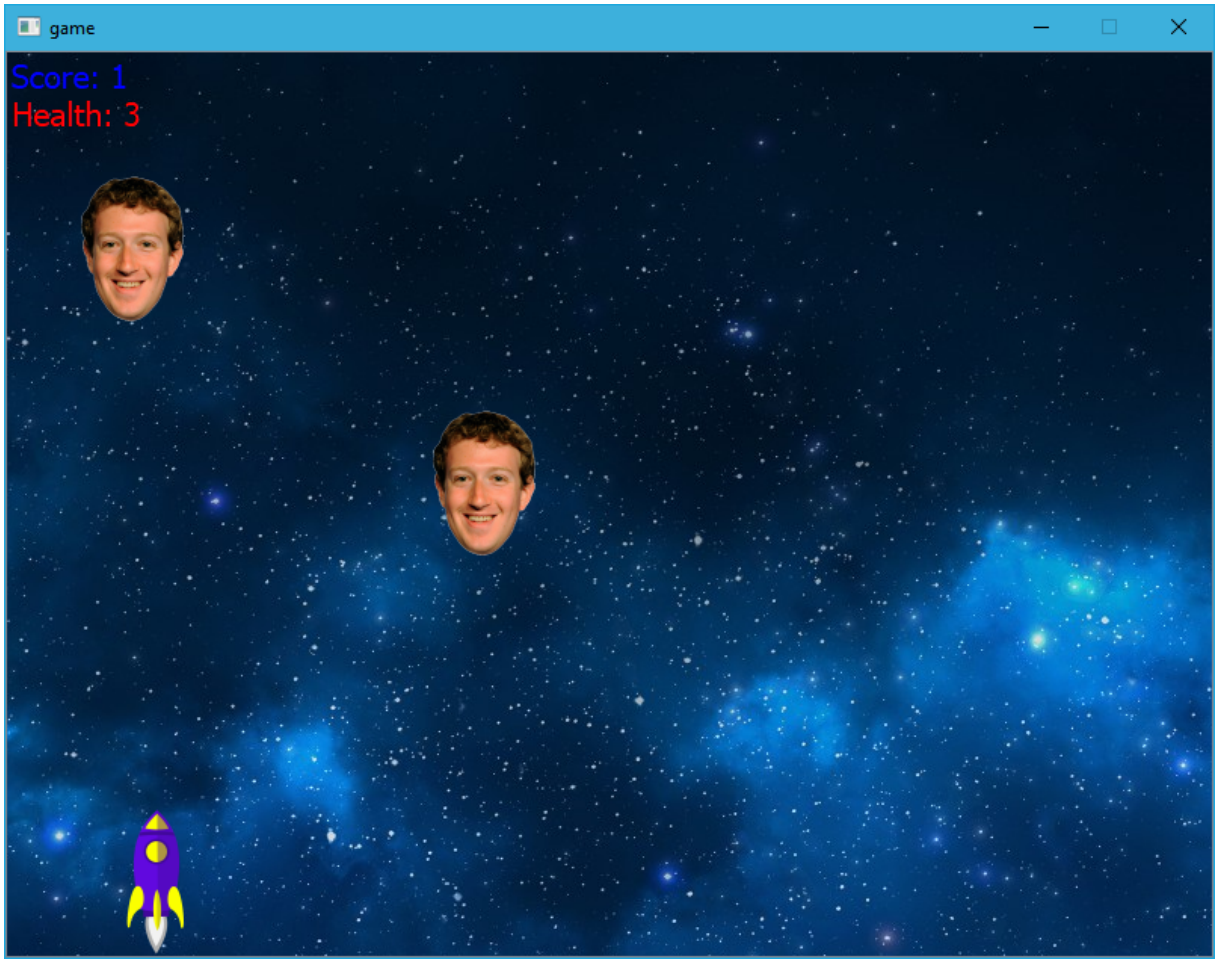

Figure 3.1– Game

4 Conclusions

[**?**] During this laboratory work i learned to how create a game using the oop principles of Inheritance, Polimorphism ,Abstraction and Encapsulation.

References

1 **OOP principles** - https://www.geeksforgeeks.org/basic-concepts-of-object-oriented-programming-using-c/

2 **Git Repository code** - https://github.com/LordOfNightmares/APPOOP/tree/master/game

3 **Tutorial followup** - https://www.youtube.com/watch?v=8ntEQpg7gck