بسمه تعالی

عنوان:

گزارشکار فاز آخر درس برنامه نویسی پیشرفته

استاد:

سرکار خانم مهندس بطحائیان

دانشجو:

نسترن منصوری

۹۶۱۲۳۵۸۰۴۱

ترم تحصیلی:

۴۰۰۱

فهرست

هدف:

هدف از پیادهسازی این فاز افزودن بخش گرافیکی به کد پیاده سازی شده در فازهای قبلی است.

توضیحات کد:

برای پیاده سازی این قسمت کتابخانه SFML استفاده شده است. علاوه بر این سه کلاس تحت نام های State، shape و playGame هم اضافه شده است که در ادامه توضیح داده خواهد شد.

کلاس state:

State.h:

```
#pragma once

#include <vector>
#include <string>
#include <array>
#ifndef STATE_H
#define STATE_H


class state
{
private:

    std::string addressOfPicture;

public:

    state(std::string);
    std::string getaddressOfPicture();

};

#endif // !STATE_H
```

در این کلاس ادرس تصاویر و جایی که هستند در متغیر خصوصی ای ذخیره می شود.

State.cpp:

```cpp
#include "state.h"

using namespace std;

state::state(std::string address)
    :addressOfPicture(address)
{
}


std::string state::getaddressOfPicture()
{
    return addressOfPicture;
}
```

در پیاده سازی ایم کلاس یک کانستراکتو رداریم که در آن با گرفتن آدرس تصویر مقدار آن را در متغیر addressOfPicture ذخیره می کند. در تابعی دیگر مقدار ذخیره شده در متغیر خصوصی addressOfPicture برمی گرداند.

کلاس Shape:

Shape.h:

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <array>
#include "state.h"
#ifndef SHAPE_H
#define SHAPE_H


class shape
{


public:

    std::string getAddressOfPic();
    state getStatenow();
protected:
    int nowState;
    std::vector<state>myState;
};

#endif // !SHAPE_H
```

همانطور که در تصویر مشخص است در این کلاس دو متغیر protected داریم که یکی از آنها برای ذخیره state عکس ها و دیگری وکتوری برای ذخیره ی stateها است.

:Shape.cpp

```cpp
#include "shape.h"


std::string shape::getAddressOfPic()
{
    return myState[nowState].getaddressOfPicture();
}


state shape::getStatenow()
{
    return myState[nowState];
}
```

در پیاده سازی کلاس shape دو تابع داریم. یکی از توابع آدرس عکس ها را برمی گرداند که نام آن getAddressOfPic است.

تابع دوم state فعلی تصاویر را فراخوانی می کند.

کلاس playGame:

playGame.h:

```
#pragma once
#include "shape.h"
#include <array>
#include <SFML\Audio.hpp>
#include <SFML\Graphics.hpp>
#include <SFML\System.hpp>
#include <SFML\Network.hpp>
#include <SFML\Window.hpp>
#ifndef PLAYGAME_H
#define PLAYGAME_H

class playGame
{
    std::array<std::array<shape*, 1>, 4> gameBackground;
    //shape* gameBackground[5][5];
public:
    void assingGameBackground();
    void statrtPage();
    void gamePage();
    void freeObj();
    ~playGame();


};

#endif // !GAMEPLAY
```

در این کلاس یک ارایه دو بعدی تعریف شده است برای قرار دادن تصاویر رنگ هایی که کاربر می تواند انتخاب کند. همچنین یکسری توابع در این کلاس موجود است که در ادامه توضیح داده شده است.

:playGame.cpp

```
#include "playGame.h"
//#include <random>

using namespace std;
using namespace sf;
int k = 0;
int n;
void playGame::assingGameBackground()
{

    string addressColor1 = "blue.png";
    myState.push_back(state(addressColor1));
    string addressColor2 = "red.png";
    myState.push_back(state(addressColor2));

    string addressColor3 = "yelow.png";
    myState.push_back(state(addressColor3));

    string addressColor4 = "green.png";
    myState.push_back(state(addressColor4));
```

اولین تابع، تابع assingGameBackground است. در قسمت اول این تابع آدرس تصاویری که به کاربر نمایش داده می شود در state ها ذخیره می شوند. که در ابتدا تصاویر رنگ ها برای کاربر نمایش داده می شود تا کاربر رنگ پارچه موردنظر خود را از بین چهار رنگ موجود انتخاب کند.

بعد از انتخاب رنگ پارچه شلوار و مانتو همان رنگ به کاربر نشان داده می شود تا طراحی مورد نظر خود را از بین آن دو انتخاب کند.

```
string addressPants = "P1.png";
myState.push_back(state(addressPants));

string addressManto = "M1.png";
myState.push_back(state(addressManto));
//-----------------------------------------------------------------
string addressColorPants1 = "P11.png";
myState.push_back(state(addressColorPants1));

string addressColorPants2 = "P12.png";
myState.push_back(state(addressColorPants2));

string addressColorPants3 = "P13.png";
myState.push_back(state(addressColorPants3));

string addressColorPants4 = "P14.png";
myState.push_back(state(addressColorPants4));

    string addressColorManto1 = "M11.png";
    myState.push_back(state(addressColorManto1));

    string addressColorManto2 = "M12.png";
    myState.push_back(state(addressColorManto2));

    string addressColorManto3 = "M13.png";
    myState.push_back(state(addressColorManto3));

    string addressColorManto4 = "M14.png";
    myState.push_back(state(addressColorManto4));
```

همانطور که میبیند آدرس تمامی تصاویر ذخیره می شوند تا در صورت انتخاب کاربر به او نمایش داده شوند.

تابع بعدی، تابع startPage می باشد.

```
void playGame::statrtPage()
{
    RenderWindow window(VideoMode(1280, 891), "Tailor Game", Style::Titlebar | Style::Close);
    Texture IMG;
    if (!IMG.loadFromFile("BGP.png"))
    {
        return;
    }
    Sprite sprite1(IMG);

    while (window.isOpen())
    {

        Event ev;
        while (window.pollEvent(ev))
        {

            switch (ev.type)
            {
            case Event::Closed:
                window.close();
                break;
            case Event::KeyPressed:
                if (ev.key.code == Keyboard::Escape)
                    window.close();
                break;
            }
if (ev.mouseButton.button == Mouse::Left)
{
    if ((ev.mouseButton.x >= 49 && ev.mouseButton.x <= 170) && (ev.mouseButton.y >= 725 && ev.mouseButton.y <= 852))
    {
        window.close();
        aboutPage();


    }
}
if (ev.mouseButton.button == Mouse::Left)
{
    if ((ev.mouseButton.x >= 554 && ev.mouseButton.x <= 731) && (ev.mouseButton.y >= 250 && ev.mouseButton.y <= 428))
    {
        window.close();
        gamePage();


    }
}
}

window.clear();
window.draw(sprite1);
window.display();
```

این تابع در ابتدا تصویر شروع بازی را نمایش می دهد. صفحه start بازی که به شکل زیر است و کاربر با کلیک کردن بر آن وارد محیط بازی می شود.

تابع gamePage یک دیگر از توابع این کلاس و درواقع اصلی ترین تابع است؛ که پیاده سازی و کلیک ردن ر.ی تصا.یر همگی در این تابع پیاده سازی شده اند.

و دوتابع اخر بازی که یکی دستراکتوری است که تابع دیگر را فراخوانی می کند.

```
void playGame::freeObj()
{
    for (int i = 0; i < 1; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << i << "  " << j << endl;
            shape* temp = gameBackground[i][j];
            delete temp;
            gameBackground[i][j] = nullptr;


        }
    }

}

playGame::~playGame()
{

    freeObj();

}
```

تابع freeObj برای رها سازی حافظه ی گرفته شده یا همان پاک سازی تصاویر از صفحه نمایش پیاده سازی شده است که آرایه gameBackground را در یک حلقه تو در تو تخلیه می کند.

Main():

```
#include "playGame.h"
#include <random>
#include <ctime>
using namespace std;


int main()
{
    //srand(time(0));
    playGame obj;
    obj.assingGameBackground();
    obj.statrtPage();

    return 0;

}
```

همانطور که مشخص است در تابع main برنامه هم تغییراتی بوجود امده به این صورت که با اجرای برنامه ایتدا یک شی از کلاس playGame ایجاد می شود سپس با این شی توبع assignGameBackground و startPage فراخوانی می شود.