```cpp

#include <iostream>
#include <thread>
#include<chrono>
#include "pac.h"
#include "ghost.h"
#include "maze.h"
#include "gameText.h"
#include "fruit.h"

using namespace std;
using namespace sf;

class Starter
{
 public:
     Event sfEvt;
     Maze maze;
     Pac<Starter>* pac;
     Ghost<Starter> *Blinky, *Pinky, *Inky,*
     thread ghStatusThread;
     thread loopThread;
     int attackInterval=15; // 15 sec
     int scaterInterval=7; // 7 sec
     int blueInterval = 6;
     int delay;
     int curTime;
```

```
    Texture backText;
    Texture backFlashText;

    Sprite backSpr;
    bool intro = false;
    bool isCollid = false;
    bool lifeWin = false;
    bool toNextLevel = false;
    GameSound *gmSound;
    GameText *gameText;
    Fruit* fruit;
    RenderWindow* window;

    Starter( RenderWindow *win, Texture* sp
    {
        cout << "Starter class OK - Start g
        window = win;
        gmSound = GameSound::getInstance();
        gameStatus = Demo;
        backText.loadFromFile("texture/map1
        backFlashText.loadFromFile("texture
        backSpr.setTexture(backText);
        maze.initMaze();
        pac = new Pac<Starter>(sprTexture,t
        Blinky = new Ghost<Starter>(sprText
        Pinky = new Ghost<Starter>(sprTextu
        Inky = new Ghost<Starter>(sprTextur
        Clyde = new Ghost<Starter>(sprTextu
        gameText = new GameText();
        fruit = new Fruit(sprTexture);

        loopThread = thread ( &Starter::loo
        while (win->isOpen())
        {
            //if (pac->dotsEat == 20) { nex
            if ( CntrGame::dotsEat == maze.
            while (win->pollEvent(sfEvt))
            {
                if (sfEvt.type == Event::Cl
                else if (sfEvt.type == Even
                {
                    if (gameStatus == Demo
                    {
                        startGame();
                    }
                    pac->rotation(sfEvt.key
                }
            }
        }
    };

   ~Starter()
   {
       if (loopThread.joinable()) { loopThr
       if (ghStatusThread.joinable()) { ghS
       delete  Blinky;
       delete Pinky;
       delete Inky;
       delete Clyde;
       //delete pac;
       //delete gmSound;
       //delete gameText;
   }

   //---
   void drawLife( RenderWindow *win)
   {
       if (pac->pacLife < 0) { return; }
       Sprite spr;
       for (int i = 0; i < pac->pacLife; i+
       {
           spr = pac->getLifeSpr();
           spr.setPosition( Vector2f(30*i,
           (*win).draw( spr );
       }
   }

   //---call from Pac class
   void setBlueGhost()
   {
       if (CntrGame::pacIsDead) { return; }
       cout << "Blue Status"<<endl;
       if (ghStatusThread.joinable()) { ghS
       sleep(milliseconds(20));
       ghostStatus = Blue;
       creatGhostThr();
       gmSound->play(GameSound::PlSound::Bl
       gmSound->stop(GameSound::PlSound::Si
   }

   //---cal from Ghost class
   void collidToPac()
   {
       CntrGame::pacIsDead = true;
       pac->pacLife--;
       isCollid = true;
       stopAll();
       if (pac->pacLife < 0)
       {
           gameOver();
       }
   }

   //--- STOP GAME
   void gameOver()
   {
       gameStatus = Demo;
       CntrGame::score = 0;
       CntrGame::level = 0;
       lifeWin = false;
       gameText->scoreTxt.setString("SCORE:
       pac->stop();
       Blinky->stop();
       Pinky->stop();
       Inky->stop();
       Clyde->stop();
       resetPacGhost();
       resetLevel();
   }

private:

   //---
   void loop( RenderWindow* win)
   {
       win->setActive(true);
       while (win->isOpen())
       {
           if (CntrGame::score >= 10000 &&
           {
               lifeWin = true;
               gmSound->play(GameSound::PlS
               pac->pacLife++;
           }
           win->clear();
```

```cpp
if (gameStatus == Play)
{
    drawLife(win);
    maze.drawWall(win);
    win->draw(backSpr);
    win->draw(gameText->gameOver
    win->draw(gameText->scoreTxt
    win->draw(gameText->levelTxt
    if (ghostStatus == Blue)
    {
        string dif = to_string(c
        if (stoi(dif) == 0) { di
        gameText->countTxt.setSt
        win->draw(gameText->coun
    }
    if (gameText->bonusTxt.getSt
    {
        win->draw(gameText->bonu
    }
    if(fruit->getVisible() ){ wi
}
else
{
    win->draw(gameText->enterTxt
    win->draw( Blinky->getNameTx
    win->draw(Pinky->getNameTxt(
    win->draw(Inky->getNameTxt()
            win->draw(Clyde->getNameTxt(
        }

        win->draw(pac->getSprite());
        win->draw(Blinky->getSprite());
        win->draw(Pinky->getSprite());
        win->draw(Inky->getSprite());
        win->draw(Clyde->getSprite());
        win->display();

    }
 }

//---
void changeGhostState()
{
           cout << "Start thread for Gho
           delay =  scaterInterval;
           if (ghostStatus == Blue)
           {
               delay = blueInterval;
               CntrGame::isBlueGhost = t
           }
           changeStatus();
           //cout << "Status=" << ghostS
           while ( ghStatusThread.joinab
           {
```

```
curTime = time(0);
curTime += delay;

while (true && ghStatusTh
{
    if (curTime <= time(0
}// wait for change ghost

sleep(milliseconds(10));
if (ghostStatus == Blue)
{
  CntrGame::isBlueGhost =
  CntrGame::ghostBonus =
  gmSound->stop(GameSound
  if(CntrGame::gameRun)gm
}
if (ghostStatus == Attack
{
    ghostStatus = Scater;
    delay = scaterInterva
    cout << "Scater " <<

}
else
{
    ghostStatus =  Attack
    delay = attackInterva
```

```
                   cout << "Attack " <<
               }
               changeStatus();

               if (( maze.dotsCount - Cn
               {
                   gmSound->setPich(Game
               }

           }
          cout << "Ended Thread GhostSta
          cout << "**********************
}

//---
void changeStatus()
{
    Blinky->changeGhostState();
    Pinky->changeGhostState();
    Inky->changeGhostState();
    Clyde->changeGhostState();
}

//---
void stopAll()
{
    pac->stop();
```

```
        Blinky->stop();
        Pinky->stop();
        Inky->stop();
        Clyde->stop();
        CntrGame::gameRun = false;
        gameText->stopThread();
        fruit->stop();
        gmSound->stopAll();
        if (ghStatusThread.joinable() ){ ghS
        resetLevel();
        wait(2);
        if (pac->pacLife >= 0) { startLevel(
    }

    //---
    void startGame()
    {
        maze.redrawDot();
        gameText->gameOverTxt.setString("");
        pac->pacLife = 2;
        gameStatus = Play;
        resetPacGhost();
        intro = true;
        CntrGame::gameRun = true;
        CntrGame::level=1;
        CntrGame::score=0;
        CntrGame::dotsEat = 0;
        blueInterval = 6;
        fruit->setLevel(CntrGame::level);
        gmSound->setPich(GameSound::Siren, 1
        startLevel();
    }

    //---
    void startLevel()
    {
        if (intro)
        {
            intro = false;
            gmSound->play(GameSound::PlSound
        }
        isCollid = false;
        ghostStatus = Scater;
        gmSound->play(GameSound::PlSound::Si
        CntrGame::gameRun = true;
        resetPacGhost();
        pac->run();
        creatGhostThr();
        fruit->start();
        gameText->levelTxt.setString("LEVEL:
    }


    //---
    void nextLevel()
    {
        gmSound->setPich(GameSound::Siren, 1
        toNextLevel = true;
        CntrGame::dotsEat = 0;
        stopAll();
        maze.redrawDot();
        CntrGame::level++;
        gameText->levelTxt.setString("LEVEL:
        fruit->setLevel(CntrGame::level);
        if (CntrGame::level > 2 && CntrGame:
        else if (CntrGame::level >= 6 && Cnt
        else if (CntrGame::level >= 10) { bl
    }

    //---
    void creatGhostThr()
    {
        while (ghStatusThread.joinable()) {}
        ghStatusThread = thread(&Starter::ch
    }

    //---
    void resetLevel()
    {
        CntrGame::ghostBonus = 100;
        if(pac->pacLife<0)
        {
            gameText->gameOverTxt.setString(
        }
    }

    //---
    void resetPacGhost()
    {
        pac->reset();
        Blinky->reset();
        Pinky->reset();
        Inky->reset();
        Clyde->reset();
    }

    // wait for second
    void wait(int delayInt)
    {
        auto curTime = time(0);
        int counter=0;
        curTime += delayInt;
        while (true)
        {
            counter++;
            if (toNextLevel)
            {
                if (counter % 30 == 0)
```

```
                    {
                     backSpr.setTexture(backFlas
                    }
                    else if (counter % 30 == 15)
                    {
                     backSpr.setTexture(backText
                    }
                }
                if (curTime < time(0)) { break;
                sleep(milliseconds(10));
            }
            toNextLevel = false;
            backSpr.setTexture(backText);
        }

};


```