Разработка системы классов для игры «Snake»

Назаров Михаил Родионович
4936

Предметная область

Змейка ест яблоко

Змейка увеличивается на один сегмент

Если змейка сталкивается с собой или со стенкой, игра прекращается

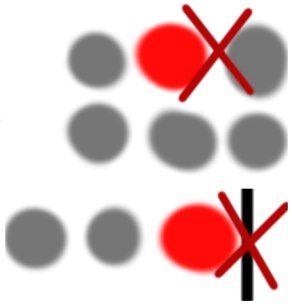
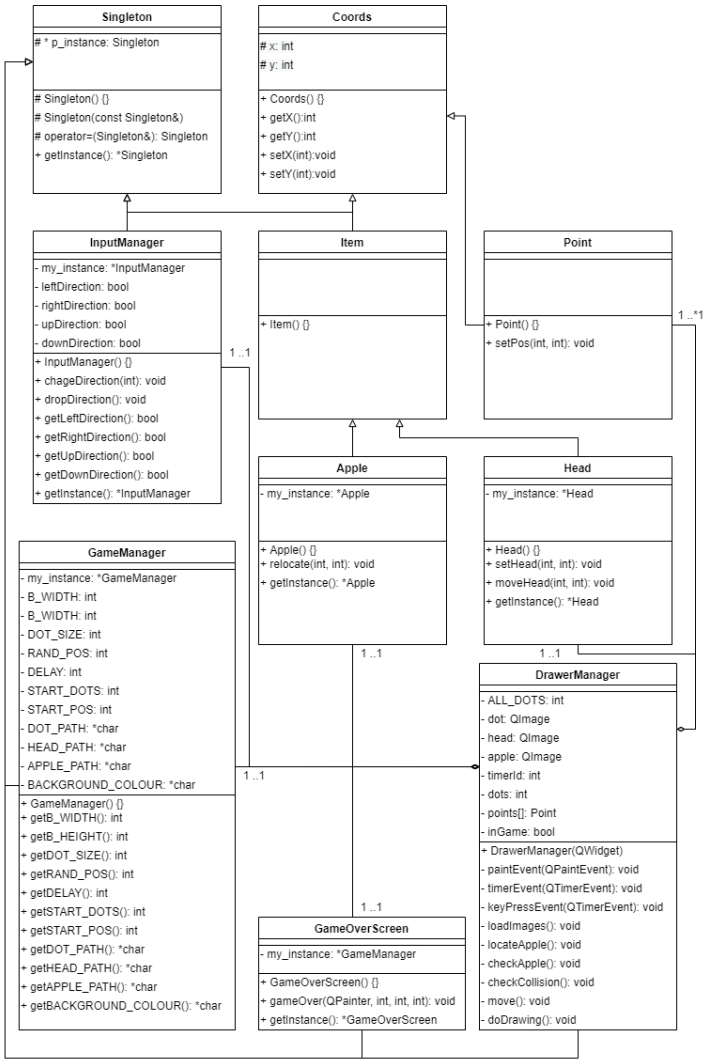


Диаграмма классов

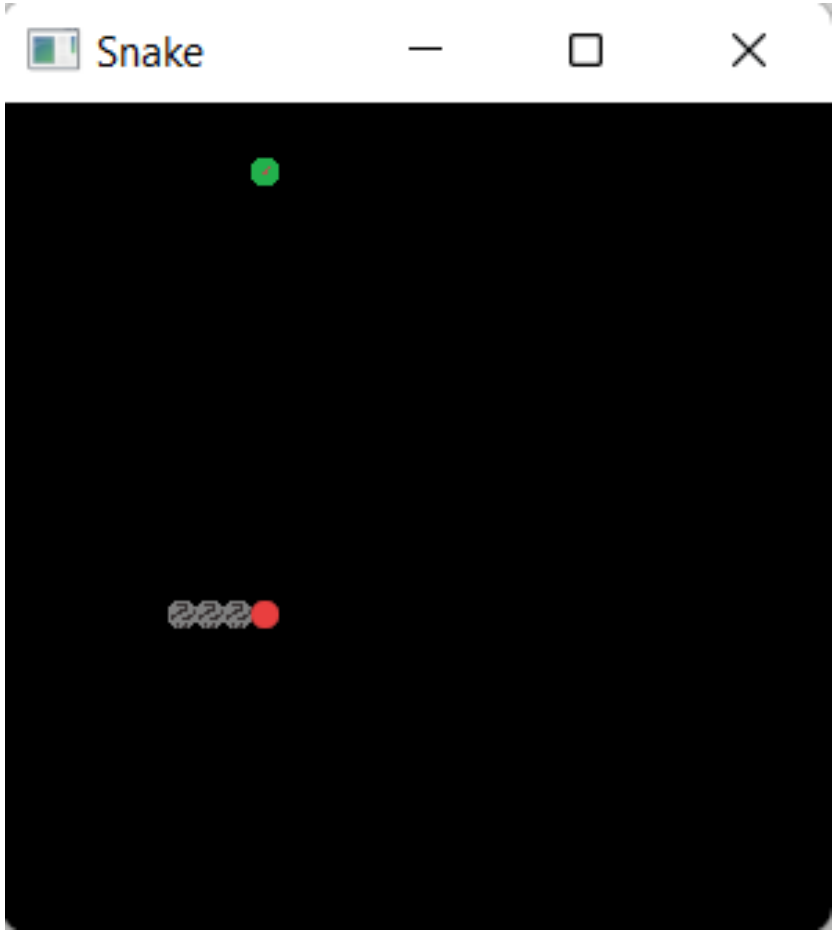


Паттерн проектирования Singleton

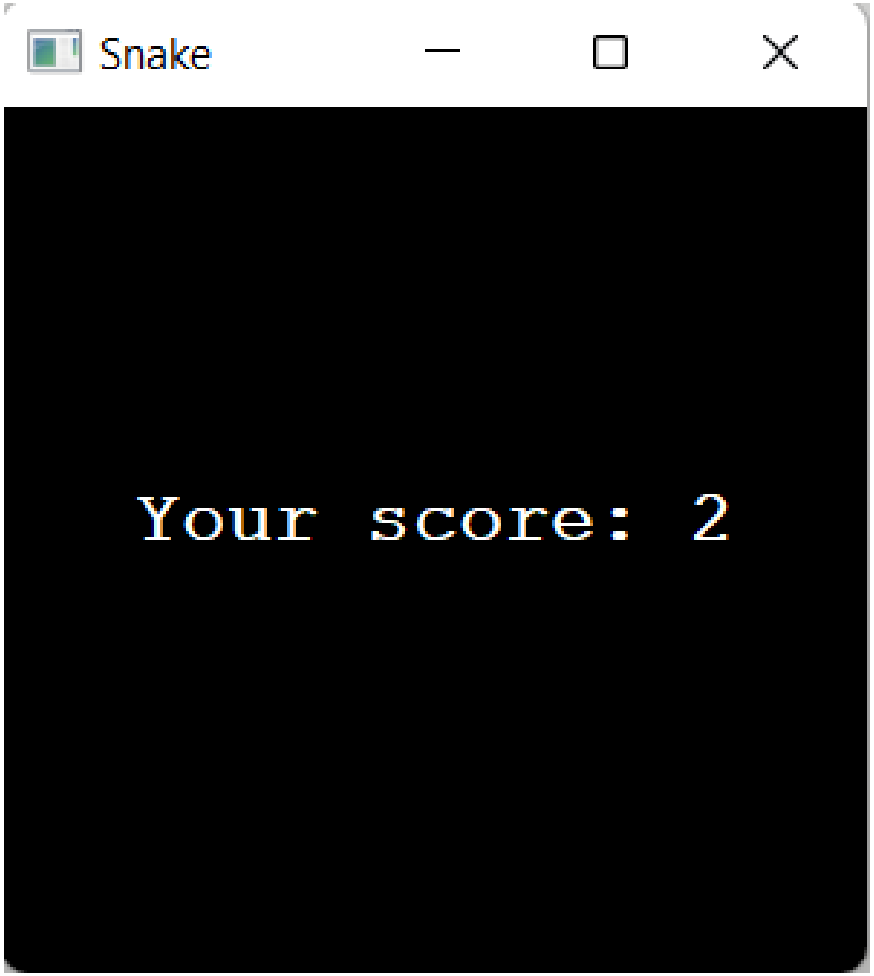
```
#ifndef SINGLETON_H
#define SINGLETON_H

//Базовый класс. Отвечает за создание и поддержание единственного экземляра класса
class Singleton
{
public:
    //Получить ссылку на единственный экземляр класса
    static Singleton * getInstance() {
        if(!p_instance)
            p_instance = new Singleton();
        return p_instance;
    }
protected:
    //Статичная ссылка на экземпляр класса
    static Singleton * p_instance;
    //Стандартный пустой конструктор
    Singleton() {}
    //Конструктор с параметром
    Singleton( const Singleton& );
    //Переопределение оператора =
    Singleton& operator=( Singleton& );
};
#endif // SINGLETON_H
```

Разработка интерфейса (игра)



Разработка интерфейса (экран конца игры)



Программирование основных методов

- Отрисовка изображений в приложении

```
//Функция отрисовки змейки и яблока
//Не принимает значений
//Не возвращает значений
void DrawerManager::doDrawing() {

    //Создать объект QPainter
    QPainter qp(this);

    //Если игра все еще идет
    if (inGame) {
        //то отрисовать яблоко
        qp.drawImage(Apple::getInstance()->getX(), Apple::getInstance()->getY(), apple);

        //и голову с точками
        for (int z = 0; z < dots; z++) {
            if (z == 0) {
                qp.drawImage(Head::getInstance()->getX(), Head::getInstance()->getY(), head);
            } else {
                qp.drawImage(points[z].getX(), points[z].getY(), dot);
            }
        }
    //иначе
    } else {
        //вывести экран конца игры
        GameOverScreen::getInstance()->gameOver(qp, height(), width(),(dots - GameManager::getSTART_DOTS()));
    }
}
```

- Перемещение змейки

```
//Функция перемещения змейки
//Не принимает занчений
//Не возвращает значений
void DrawerManager::move() {
   //Создать значения для смещения змейки
   int offsetX = 0;
   int offsetY = 0;
   //Переместить каждую точку на координату следующей точки
   for (int z = dots; z > 0; z--) {
      if(z == 1)
      {
         points[z].setPos(Head::getInstance()->getX(),Head::getInstance()->getY());
      }
      else
      {
         points[z].setPos(points[(z-1)].getX(), points[(z-1)].getY());
      }

   }

   //Если текущее направление - влево
   if (InputManager::getInstance()->getLeftDirection()) {
      //то добавить смещение влево
      offsetX += GameManager::getDOT_SIZE();
   }
   //Если текущее направление - вправо
   if (InputManager::getInstance()->getRightDirection()) {
      //то добавить смещение вправо
      offsetX -= GameManager::getDOT_SIZE();
   }
   //Если текущее направление - вверх
   if (InputManager::getInstance()->getUpDirection()) {
      //то добавить смещение вверх
      offsetY += GameManager::getDOT_SIZE();
   }
   //Если текущее направление - вниз
   if (InputManager::getInstance()->getDownDirection()) {
      //то добавить смещение вниз
      offsetY -= GameManager::getDOT_SIZE();
   }

   //Переместить голову на указанное смещение
   Head::getInstance()->moveHead(offsetX,offsetY);
}
```

- Столкновение змейки с собой или границей экрана

```
//Функция проверки столкновения змейки со стеной или собой
//Не принимает значений
//Не возвращает значений
void DrawerManager::checkCollision() {

   //Проверить каждую точку змейки
   for (int z = dots; z > 0; z--) {

      //Если координаты головы и точки совпадают
      if ((z > GameManager::getSTART_DOTS() + 1) && (Head::getInstance()->getX() == points[z].getX()) && (Head::getInstance()->getY() ==
    points[z].getY())) {
         //то закончить игру
         inGame = false;
      }
   }

   //Если координаты головы вышли за рамки экрана
   if (Head::getInstance()->getY() >= GameManager::getB_HEIGHT() ||
         Head::getInstance()->getY() < 0 ||
         Head::getInstance()->getX() >= GameManager::getB_WIDTH()||
         Head::getInstance()->getX() < 0) {
      //то закончить игру
      inGame = false;
   }

   //Если игра закончена
   if(!inGame) {
      //то уничтожить таймер
      killTimer(timerId);
   }
}
```

- Столкновение змейки с яблоком

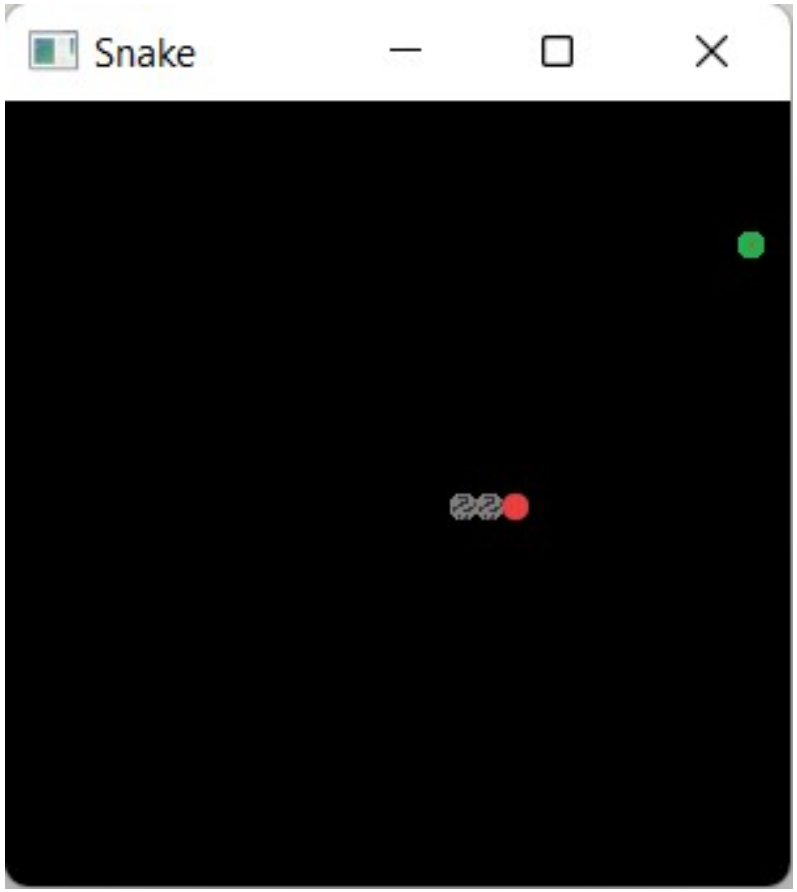
```
//Функция проверки столкновения змейки с яблоком
//Не принимает значений
//Не возвращает значений
void DrawerManager::checkApple() {

    //Если координаты головы и яблока совпадают
    if (((Head::getInstance()->getX()) == Apple::getInstance()->getX()) && (Head::getInstance()->getY())
     == Apple::getInstance()->getY()) {
        //то добавить змейке точку
        dots++;
        //и переместить яблоко
        locateApple();
    }

    //Если количество точек змейки больше или равно максимальному
    if (dots >= ALL_DOTS)
    {
        //то завершить игру
        inGame = false;
    }
}
```

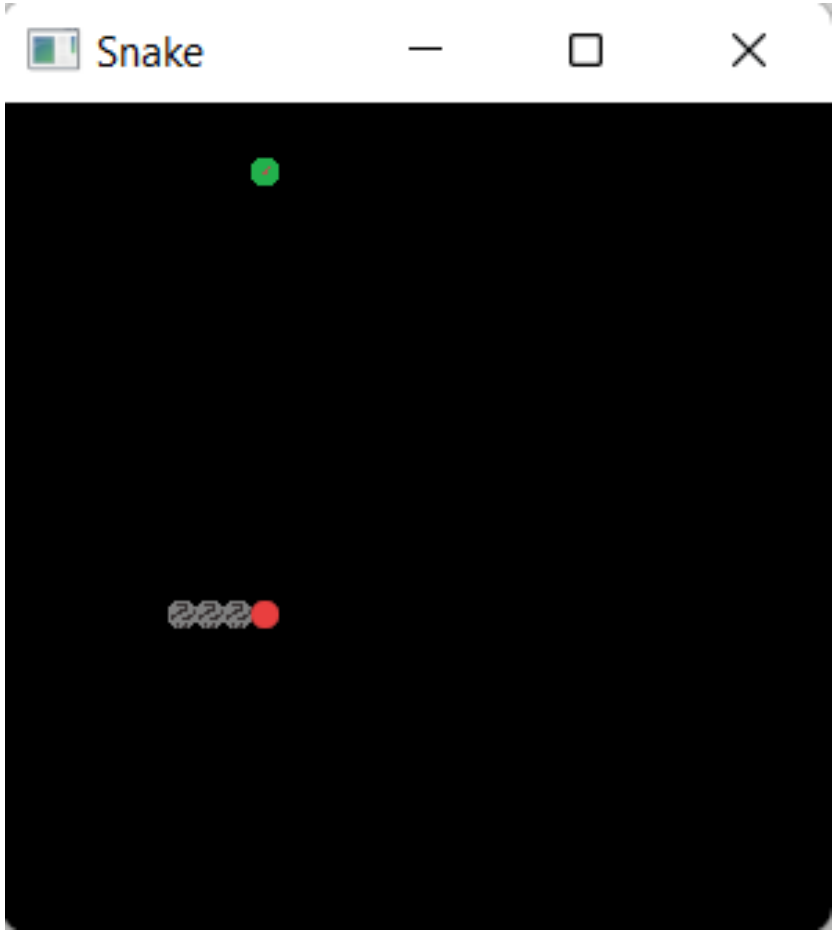
Тестирование

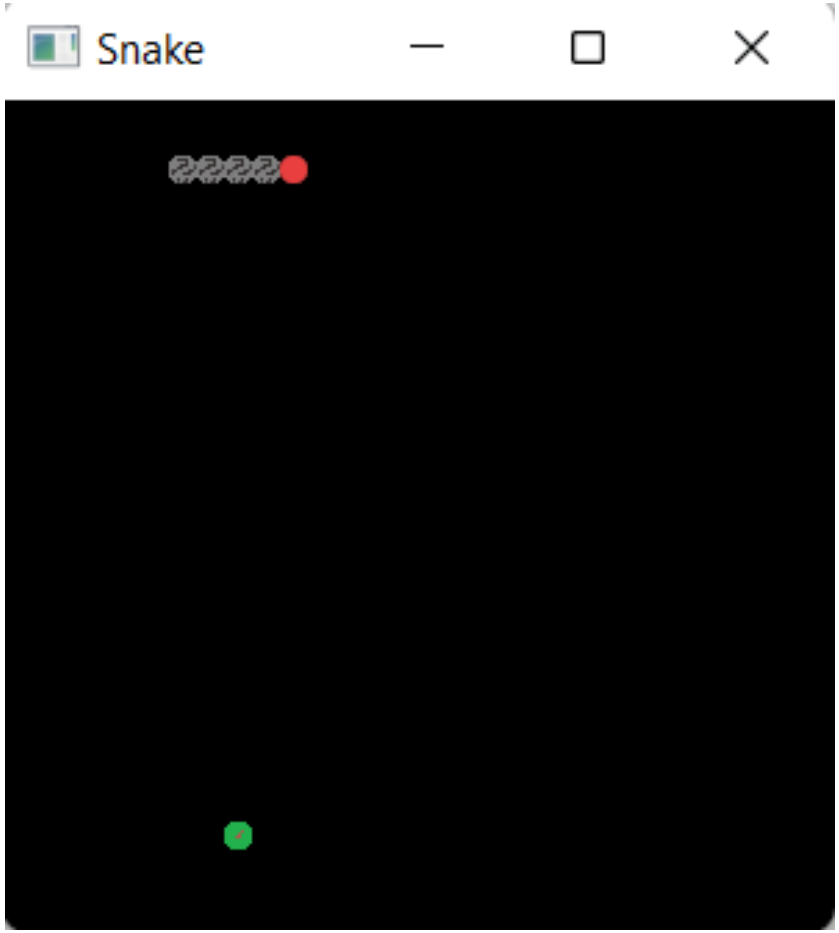
- Отрисовка игры



Тестирование

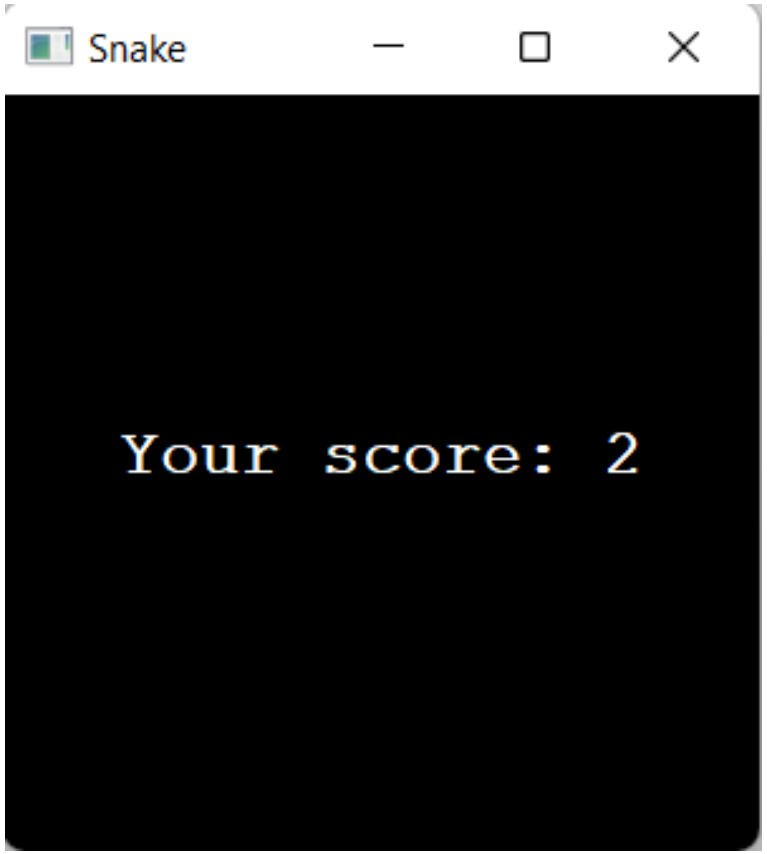
- Столкновение змейки с яблоком





Тестирование

- Столкновение головы змейки с ее телом



Тестирование

- Столкновение змейки с границей экрана