Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Программирование

Отчет по курсовой работе
Игра "Змейка"

Работу выполнил:
Ильин А.А.
Группа:
23501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2017

Содержание

1 Игра ”Змейка”

Классчиескую игру "Змейка"знают, наверное, все население нашей планеты. Простая, интересная игра потеряла свою былую популярность. Было решено создать приложение для игры "Змейка"

1.1 Краткое описание игры

Суть игры Змейка заключается в том, чтобы заполнить игровое поле только телом змейки. Она увеличивает свой размер поедая фрукты, разбросанные по полю.

1.2 Правила игры

- Цель игры.
  Заполнить игровое поле телом змейки.
- Ход игры.
  Змейка может двигаться во всех направлениях и поедать фрукты, избегая препятствий
- Окончание игры.
  Игрок проигрывает, если змейка "съедает саму себя"или сталкивается с границей поля.

1.3 Задание

Разработать приложение, позволяющее играть в игру "Змейка"одному игроку.

1.4 Вывод

Определены правила игры "Змейка которую планируется реализовать.

2 Проектирование приложения, реализующего игру ”Змейка”

Приложение должно позволять играть в "Змейку"

3 Реализация приложения ”Змейка”

3.1 Среда разработки

Интегрированная среда разработки IntelliJ IDEA 2016.2.5.
Язык: Java 1.8.
Система автоматической сборки: Gradle 2.14.

3.2 Проектирование библиотеки

Библиотека - ядро приложения. Здесь содержатся основные классы, необходимые для представления игры. Для создания графического приложения была выбрана библиотека LWJGL.

3.3 Реализация библиотеки

Основные классы, выделенные в библиотеке:

- Класс Cell. Содержит в себе информацию о клетке, реализует методы доступа к информации о клетке и счетчик "горения"клетки.
- Класс Constants. Класс содержит в себе константы для настройки приложения, инициализации графического интерфейса.

3.4 Реализация графического приложения

Для создания графического приложения была выбрана библиотека LWJGL

На рисунке 1 представлено главное окно приложения. На экране разбросаны различные фрукты.

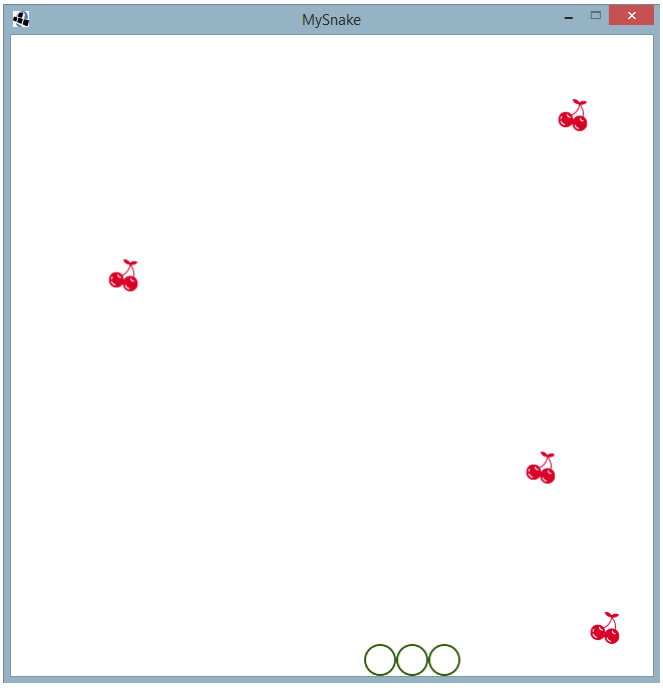

Рис. 1: Графическое приложение.

На рисунке 2 - окно игры. Съедено некоторое число фруктов. Демонстрируется разнообразие объектов.

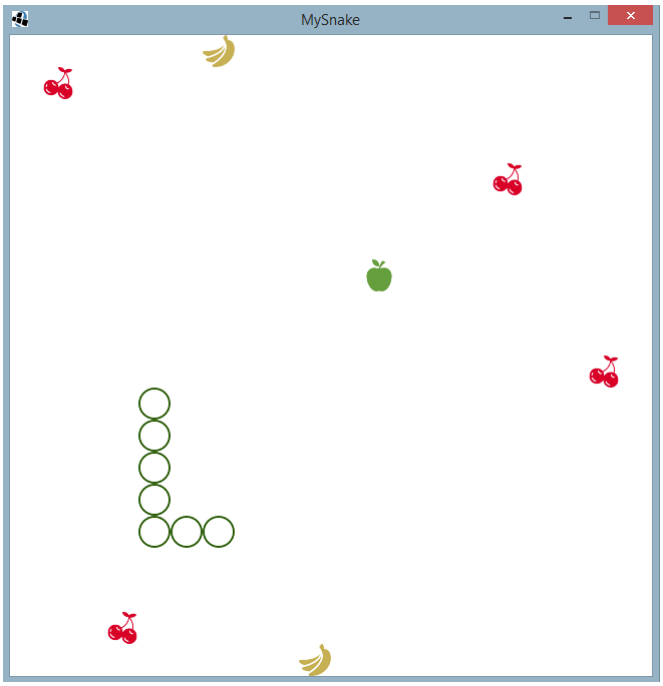

Рис. 2: Скриншот окна игры.

Таким образом, разработано графическое приложение, позволяющее играть в игру "Змейка".

3.5 Вывод

Для реализации игры определены основные классы библиотеки, графического приложений. Разработано графическое приложение, предоставляющее возможность играть в "Змейку".

4 Вывод

В результате работы над проектом была разработана библиотека для игры "Змейка". Посредством библиотеки LWJGL было создано приложение, предоставляющее пользователю функциональность ядра и позволяющее играть в игру "Змейка".

5 Приложение 1. Листинги кода

5.1 Библиотека

```
1  package Core;
2
3  import GUI.Sprite;
4
5  import static Core.Constants.CELL_SIZE;
6
7
8  public class Cell {
9
10     private int x;
11     private int y;
12     private int state;
13
14     public Cell (int x, int y, int state){
15         this.x=x;
16         this.y=y;
17         this.state=state;
18     }
19
20     public int getX(){
21         return x;
22     }
23
24     public int getY(){
25         return y;
26     }
27
28     public int getHeight(){
29         return CELL_SIZE;
30     }
31
32     public int getWidth(){
33         return CELL_SIZE;
34     }
35
36     public int getState(){
37         return this.state;
38     }
39
40     public void setState(int state){
41         this.state = state;
42     }
43
44     public void update(boolean have_to_decrease){
45         if (have_to_decrease && this.state>0){
46             this.state--;
47         }
48     }
49
50
51     public GUI.Sprite getSprite(){
52         if(this.state>0){
53
54             return GUI.Sprite.BODY;
55         }else if(this.state==0){
56
57             return null;
58         }else{
```

```
59              switch(this.state){
60                  case -1: return Sprite.CHERRIES;
61                  case -2: return Sprite.APPLE;
62                  default: return Sprite.BANANAS;
63              }
64
65
66              }
67          }
68      }
```

```java
package Core;


public class Constants {
    public static final int CELL_SIZE = 32;

    public static final int CELLS_COUNT_X = 20;
    public static final int CELLS_COUNT_Y = 20;

    public static final int INITIAL_SPAWN_CHANCE = 1;

    public static final int FPS = 5;

    public static final int SCREEN_WIDTH =CELLS_COUNT_X*CELL_SIZE;
    public static final int SCREEN_HEIGHT = CELLS_COUNT_Y*CELL_SIZE
    ↪ ;
    public static final String SCREEN_NAME = "MySnake";
}
```

5.2 Графическое приложение

```
1  package GUI;
2
3  import org.lwjgl.LWJGLException;
4  import org.lwjgl.opengl.Display;
5  import org.lwjgl.opengl.DisplayMode;
6
7  import java.util.Random;
8  import Core.Cell;
9
10 import static Core.Constants.*;
11 import static org.lwjgl.opengl.GL11.*;
12
13
14 public class GUI {
15
16     private static Cell[][] cells;
17
18
19     public static void init(){
20         initializeOpenGL();
21
22         cells = new Cell[CELLS_COUNT_X][CELLS_COUNT_Y];
23
24         Random rnd = new Random();
25
26         for(int i=0; i<CELLS_COUNT_X; i++){
27             for(int j=0; j<CELLS_COUNT_Y; j++){
28                 cells[i][j]=new Cell(i*CELL_SIZE, j*CELL_SIZE,rnd.
    ↪ nextInt(100)<INITIAL_SPAWN_CHANCE?-1:0);
29
30             }
31         }
32     }
33
34
35     public static void update(boolean have_to_decrease) {
36         updateOpenGL();
37
38         for(Cell[] line:cells){
39             for(Cell cell:line){
40                 cell.update(have_to_decrease);
41             }
42         }
43     }
44
45     public static void draw(){
46
47         glClear(GL_COLOR_BUFFER_BIT);
48
49         for(Cell[] line:cells){
50             for(Cell cell:line){
51                 drawElement(cell);
52             }
53         }
54     }
55
56     private static void drawElement(Cell elem){
57         if(elem.getSprite()==null) return;
58         elem.getSprite().getTexture().bind();
59
```

```
60          glBegin(GL_QUADS);
61          glTexCoord2f(0,0);
62          glVertex2f(elem.getX(),elem.getY()+elem.getHeight());
63          glTexCoord2f(1,0);
64          glVertex2f(elem.getX()+elem.getWidth(),elem.getY()+elem.
        ↪ getHeight());
65          glTexCoord2f(1,1);
66          glVertex2f(elem.getX()+elem.getWidth(), elem.getY());
67          glTexCoord2f(0,1);
68          glVertex2f(elem.getX(), elem.getY());
69          glEnd();
70      }
71
72      public static int getState(int x, int y){
73          return cells[x][y].getState();
74      }
75
76      public static void setState(int x, int y, int state){
77          cells[x][y].setState(state);
78      }
79
80
81      private static void updateOpenGL() {
82          Display.update();
83          Display.sync(FPS);
84      }
85
86      private static void initializeOpenGL(){
87          try {
88
89              Display.setDisplayMode(new DisplayMode(SCREEN_WIDTH,
        ↪ SCREEN_HEIGHT));
90
91
92              Display.setTitle(SCREEN_NAME);
93
94
95              Display.create();
96          } catch (LWJGLException e) {
97              e.printStackTrace();
98          }
99
100         glMatrixMode(GL_PROJECTION);
101         glLoadIdentity();
102         glOrtho(0,SCREEN_WIDTH,0,SCREEN_HEIGHT,1,-1);
103         glMatrixMode(GL_MODELVIEW);
104
105
106         glEnable(GL_TEXTURE_2D);
107         glEnable(GL_BLEND);
108         glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
109
110         glClearColor(1,1,1,1);
111     }
112 }
```

```java
package GUI;

import Core.Constants;


import org.lwjgl.input.Keyboard;
import org.lwjgl.opengl.Display;

import java.util.Random;

import static Core.Constants.*;

public class Main {
    private static boolean isExitRequested=false;
    private static int x=0,y=0, direction=1, length=3;
    private static boolean have_to_decrease = true;

    public static void main(String[] args) {
        GUI.init();
        generate_new_obj();

        while(!isExitRequested){
            input();

            move();

            GUI.draw();
            GUI.update(have_to_decrease);

        }
    }

    private static void move() {
        have_to_decrease=true;

        switch(direction){
            case 0:
                y++; break;
            case 1:
                x++; break;
            case 2:
                y--; break;
            case 3:
                x--; break;
        }

        if(x<0 || x>= Constants.CELLS_COUNT_X || y<0 || y>=
    ↪ Constants.CELLS_COUNT_Y){

            System.exit(1);
        }

        int next_cell_state = GUI.getState(x,y);

        if(next_cell_state>0){

            System.exit(1);

        }else{
            if(next_cell_state<0){
                length++;
                generate_new_obj();
```

```
62                  have_to_decrease=false;
63              }
64              GUI.setState(x,y,length);
65          }
66      }
67
68      private static void generate_new_obj() {
69          int rndPoint = new Random().nextInt(Constants.CELLS_COUNT_X
    ↪ *Constants.CELLS_COUNT_Y-length);
70
71          for(int i=0; i<CELLS_COUNT_X; i++){
72              for(int j=0; j<CELLS_COUNT_Y; j++){
73                  if(GUI.getState(i,j)<=0) {
74                      if (rndPoint == 0) {
75                          GUI.setState(i, j, -(new Random().nextInt
    ↪ (3)+1));
76                          return;
77                      }else {
78                          rndPoint--;
79                      }
80                  }
81              }
82          }
83
84      }
85
86
87      private static void input(){
88          while(Keyboard.next()){
89              if(Keyboard.getEventKeyState()){
90                  switch(Keyboard.getEventKey()) {
91                      case Keyboard.KEY_ESCAPE:
92                          isExitRequested = true;
93                          break;
94                      case Keyboard.KEY_UP:
95                          if(direction!=2) direction=0;
96                          break;
97                      case Keyboard.KEY_RIGHT:
98                          if(direction!=3) direction=1;
99                          break;
100                     case Keyboard.KEY_DOWN:
101                         if(direction!=0) direction=2;
102                         break;
103                     case Keyboard.KEY_LEFT:
104                         if(direction!=1) direction=3;
105                         break;
106                 }
107             }
108         }
109
110
111     }
112
113 }
```

```
1  package GUI;
2  import org.newdawn.slick.opengl.Texture;
3  import org.newdawn.slick.opengl.TextureLoader;
4
5  import java.io.File;
6  import java.io.FileInputStream;
7  import java.io.IOException;
8
9  public enum Sprite {
10     BODY("circle"), CHERRIES("cherries"), APPLE("apple"), BANANAS("
       ↪ bananas");
11
12     private Texture texture;
13
14     private Sprite(String texturename){
15         try {
16             this.texture = TextureLoader.getTexture("PNG", new
       ↪ FileInputStream(new File("res/"+texturename+".png")));
17         } catch (IOException e) {
18             e.printStackTrace();
19         }
20
21     }
22
23     public Texture getTexture(){
24         return this.texture;
25     }
26 }
```