# Snake Remake

## Requirements

### User inputs

The user can start and paus the program by pressing specific keys. Therefore two methods are needed, one to react on the keystroke S to start it and one for pausing the game reacting on keystroke P .

The user can change the direction of the snake by using the arrow keys. To fulfill this task a method has to handle the direction changes. It makes sure that you are not able to go in the opposite direction as the current one. This method reacts on keystrokes: ← to go left, → to go right, ↑ to go up and ↓ to move down.

The third possible user input is to change the color of the snake by pressing the C key: The method reacts on the keystroke and changes the color of the snake from black to red.

### GameGrid

The user interface in our program is based on a JGameGrid library which is made for creating user interfaces in object oriented programming. It helps creating the window frame and how the snakes moves around. The JGameGrid methods define the size and color of the game field as well as the snake. One method is responsible for showing the snake’s movement and another make apples popping up on random spots on the field. The Panel also includes labels which show the current score and the timer.

### Snake

The snake movements have to follow certain rules which must be checked by a method which stop the program if the snake reaches either the border or hits its own body. Another method keeps track on the apples which have been eaten and make sure that the body of the snake becomes longer with each piece. The snake moves at a current speed in the current direction till it breaks a rule. Therfore a method is needed which moves all body points of the snake one “pixel” further every 100 milliseconds. The speed of the snake begins to increase after a certain amount of time. (Level system: After every 30 seconds you reach a higher level of difficulty and the snake moves faster.)

### General

We need a method which creates the score by adding points by each apple which have been eaten. A clock method keeps track of how long the user has been playing.

After the game is over a method would print out a statement “GAME OVER” with the user’s score and how long he survived in the game. An optional method would be to keep track on the score and time and save its values in an array to show later on how much the user has improved and what is his highest score.

## Timetable and Milestones

The following table shows the required milestones and self-defined subtasks to ensure the success of the project.

### Planning Papers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | due to | Content | Responsible person | ✓ |
| **Requirements Paper** | 10/20 | Describe program requirements. | **Mike (pdf by email)** |  |
| 10/17 | Write prose, create table | Angi | ✓ |
| 10/19 | IEEE format | Angi |  |
| 10/17 | General behavior of the program | Angi / Anthony | ✓ |
| 10/17 | User inputs | Angi / Anthony | ✓ |
| 10/17 | Describing milestones | Angi / Anthony | ✓ |
| 10/19 | Check spelling, finish it up | Mike |  |
|  |  |  |  |  |
| **Program-Design Paper** | ??? | **Describe the design of the project implementation.** The description should include English prose, UML diagrams and any viewgraphs necessary to elucidate the design. | **Angi (send pdf by email)** |  |
|  | Define structure | all |  |
|  | Define classes |  |  |
|  | Define methods |  |  |
|  | Define relationships between classes in prose |  |  |
|  | UML-Diagramm |  |  |
|  |  |  |  |
|  | Draw Viewgraphs | Angi |  |
|  | Finish: Put everything together in IEEE format | Angi |  |
|  | Check spelling | Anthony |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iteration 1: |  |  |  | |  | |
| **Subtasks for Iteration 1** | 10/26 | Read Snake & GameGrid Tutorials | | all | |  | |
|  | Document your sources and your progess | | all | |  | |
| 10/27 | Try to run and install GameGrid class | | Angi (selfstudy) | |  | |
| 10/27 | Try to create a game field | | Angi (selfstudy) | |  | |
| 10/27 | Study the math method for random apple method | | Angi (selfstudy) | |  | |
| 10/27 | Look over logic of program | | Anthony | |  | |
| 10/27 | Study/code methods for snake body method and try to write its method | | Anthony | |  | |
| 10/27 | Try out how the the JGameGrid reactes on keystrokes | | Mike | |  | |
| 10/27 | How do we check the user inputs to be valid | | MIke | |  | |
|  |  | |  | |  | |
|  |  |  | |  | |  | |
| **Iteration 1** | **10/30** | **First round of implementing and testing.** (Write and test individual functions of the program) | | **Anthony (email pdf)** | |  | |
| 10/29 | Write a GameGrid Test class and creat a game field for our Snake program | | Angi | |  | |
| 10/29 | Write the method for creating apples which are popping up on random spots | | Angi | |  | |
| 10/29 | Wirte the method to form the snake | | Anthony | |  | |
| 10/29 | Make it be able to move around | | Anthony | |  | |
| 10/29 | Write test-program which reads input from the keyboard | | Mike | |  | |
| 10/29 |  | | Mike | |  | |
| 10/29 | Paper in prose which describes the progress | | all | |  | |
|  |  |  |  | |  | | | |
|  |  |  |  | |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteration 2:** |  |  |  |  |
| **Subtasks for Iteration 2** |  |  | Angi |  |
|  |  | Angi |  |
|  |  | Anthony |  |
|  |  | Anthony |  |
|  |  | Mike |  |
|  |  | MIke |  |
|  |  |  |  |
|  |  |  |  |  |
| **Iteration 2** | **11/14** | **Second round of implementing and testing.**  (First runnable Prototype) | ?? (send email to Prof) |  |
|  | Put the single pieces together and make the snake working on the JGameGrid |  |  |
|  | All user input can be read: |  |  |
|  | The snakes moves according to the keystrokes -> / <- / v / ^ |  |  |
|  | The color of the snake can be changed by pressing C. |  |  |
|  | The program can be paused by P. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Iteration 3:** |  |  |  |  |
| **Subtasks for Iteration 3** |  |  | Angi |  |
|  |  | Angi |  |
|  |  | Anthony |  |
|  |  | Anthony |  |
|  |  | Mike |  |
|  |  | MIke |  |
|  |  |  |  |
|  |  |  |  |  |
| **Iteration 3** | 12/5 | **Third round of implementing and testing.** The game should work by now. (Final Prototype) | ?? (send email to Prof) |  |
|  | All classes work together |  |  |
|  | The program can read all user inputs which are defined in the Requirements Paper |  |  |
|  | Test the program to find identify bugs in the program |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Final presentation** |  |  |  |  |
| **Subtasks for final Paper and Presentation** |  |  | Angi |  |
|  |  | Angi |  |
|  |  | Anthony |  |
|  |  | Anthony |  |
|  |  | Mike |  |
|  |  | MIke |  |
|  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Final Iteration & Presentation** | 12/9 | Final Iteration & Presentation  All bugs are eliminated and the final Project Documentation-Paper is written in IEEE format | ?? (send email to Prof and print it out) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | PPP inclusive Live-Demo of the programm | Angi (in class) |  |
|  |  |  |  |  |
|  |  |  |  |  |