Third assignment

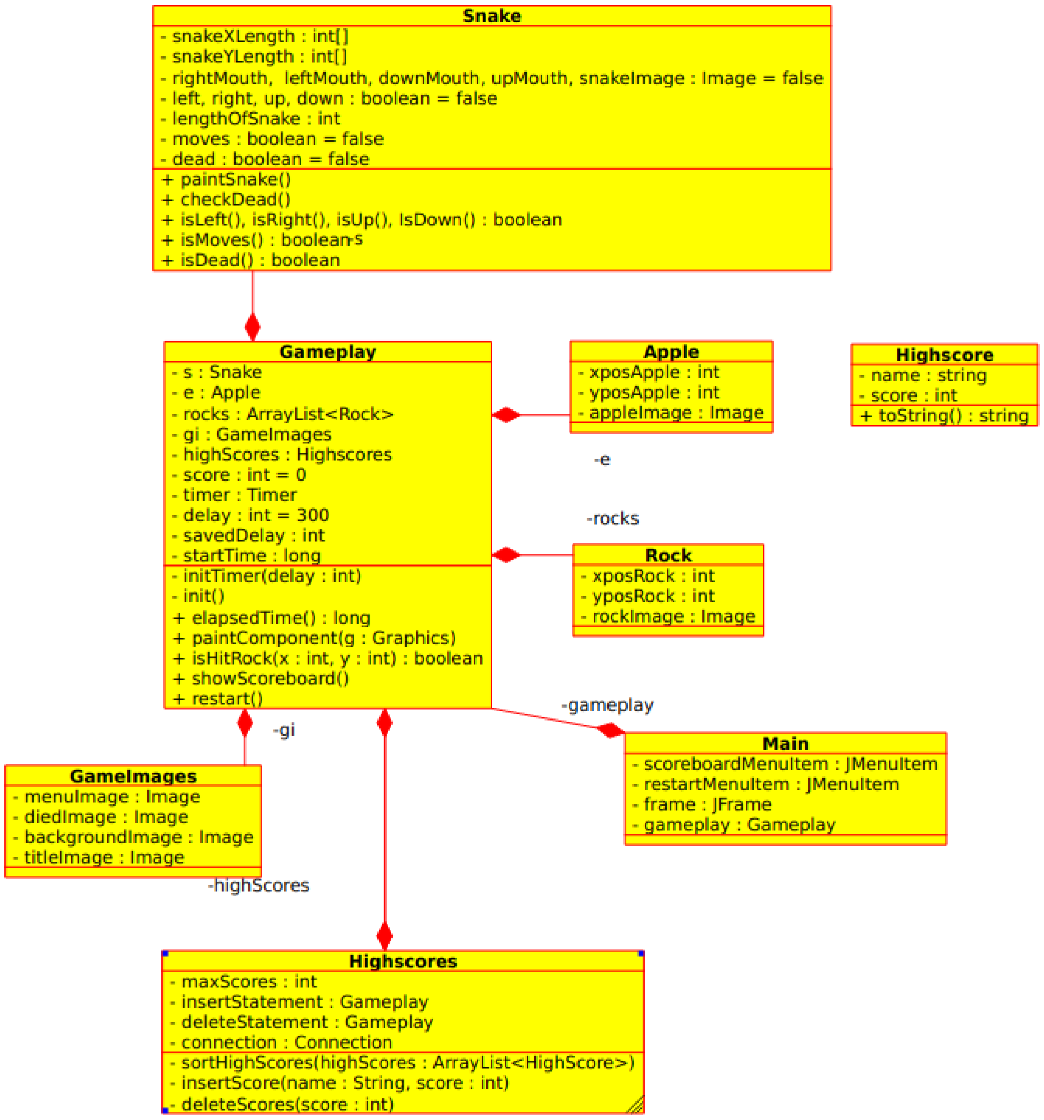
Student name: Khongorzul Khenchbish
Student id: r8hp78

Description:

2. Snake

We have a rattlesnake in a desert, and our snake is **initially two units long** (head and rattler). We have to collect with our snake the **foods on the level that appears randomly**. **Only one** food piece is placed randomly at a time on the level (on a field, where there is no snake).
The snake starts off from the center of the level in a random direction. The player can control the movement of the snake's head with keyboard buttons. **If the snake eats a food piece, then its length grows by one unit.**
It makes the game harder because there are rocks in the desert. If the snake **collides with a rock, then the game ends.** We also lose the game, if the **snake goes into itself**, or into the **boundary of the game level**.
In these situations show a popup message box, where the player can type his name and **save** it together with the amount of food eaten to the **database**. Create a menu item, which displays a highscore table of the players for the **10 best scores**. Also, create a menu item which **restarts** the game.

Class diagram:



Short description of each methods:

Snake class:

1. ***paintSnake()***
   Assigns the snake’s head into Right, Left, Top, Down direction and draws on the frame.

2. ***checkDead(), isDead(),***
   returns the status of the game.

3. ***isLeft, isRight, isUp, isDown()***
   returns the direction the snake is headed before.

4. ***isMoves()***
   Returns true if the snake started to move/game started.
   Returns false otherwise.

Gameplay class:

1. ***initTimer()***
   starts the timer(java.lang.timer) with a given delay.

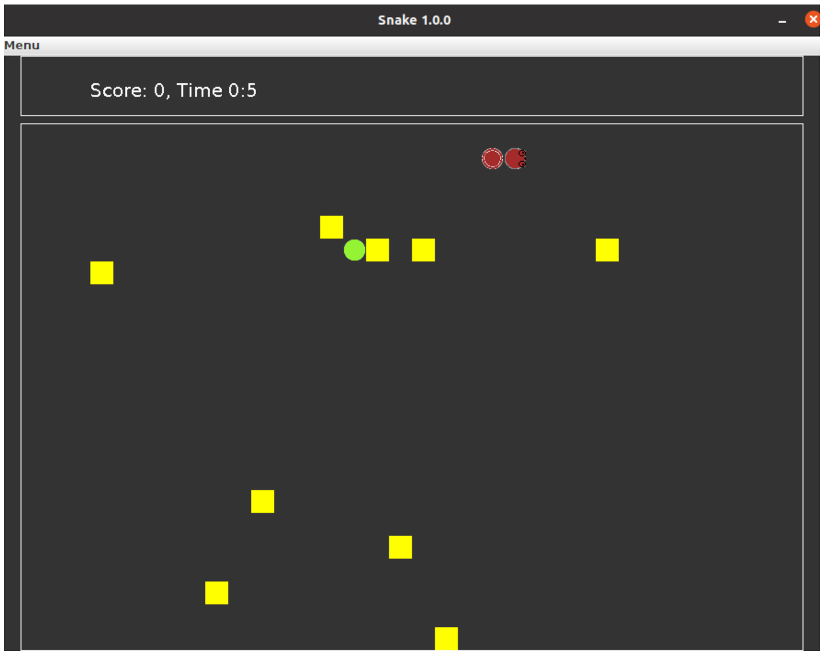
2. ***init()***
   When we start the game, or die, we call the init function to put everything back to its starting position. It lets the player choose the difficulty level and prepares the board accordingly
3. ***paintComponent()***
   - called on update of the snake move, each time,
   - draws the window from beginning starting with the header and body part of the application.
   - puts the food randomly each time whereas the rocks are in the same position
   - draws when the snake eats the food, hits the rock or hits the wall as well.
   - display game duration and player score as well
4. ***isHitRock()***
   Returns boolean value, the snake hits rock only when their position intersects.
5. ***keyPressed()***
   event listener on the up, down, left, right arrows. When the listener is activated it changes the snake head direction into the called direction.
6. ***actionPerformed()***
   keeps track of the length of the snake in all directions and modify the class variable when the snake hits the wall.
7. ***showScoreboard()***
   retrieves the table sorted by the score and name from the database and shows top 10 scored players on a board.
8. ***restart()***
   save the player's username with the achieved score into a database table then starts the game again.

Highscores class:

1. ***sortHighScores()***
   returns highscores from the database table(name, score) as sorted.
2. ***insertScore()***
   insert single row into the table containing name and score information
3. ***deleteScore()***
   delete the row from the database table where the score equal to is given parameter.
4. ***getHighScores()***
   queries the database table and returns all the rows in the table as sorted.
5. ***putHighScore()***
   if the number of data is less than 10, it adds the info to the table. If not, it compares the current score with the least one and replaces the greater one into the table.

The testing:

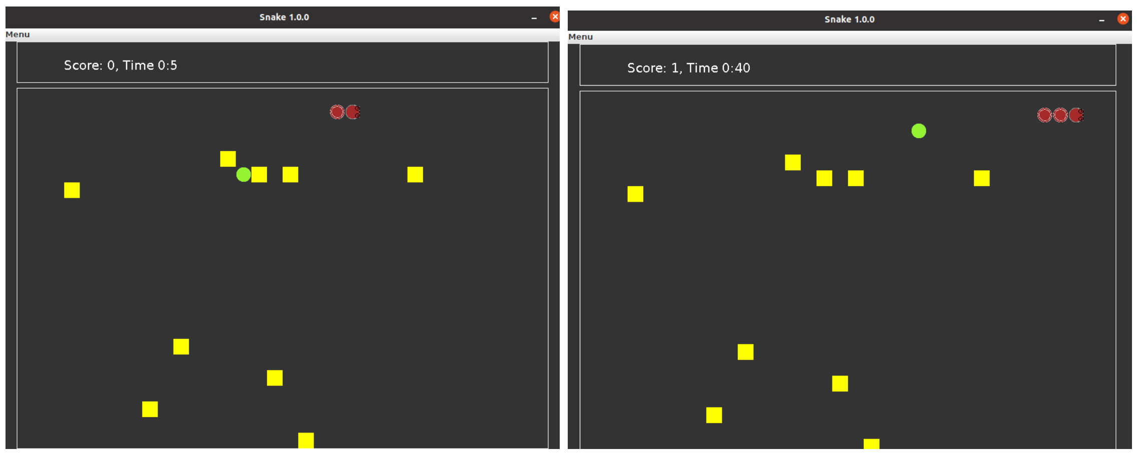
1. ***Initial length f the snake is 2(head and rattler)***



Score: 0, Time 0:5

fig 1.

2. ***Food should appear randomly while it's not true for rocks***



Score: 0, Time 0:5

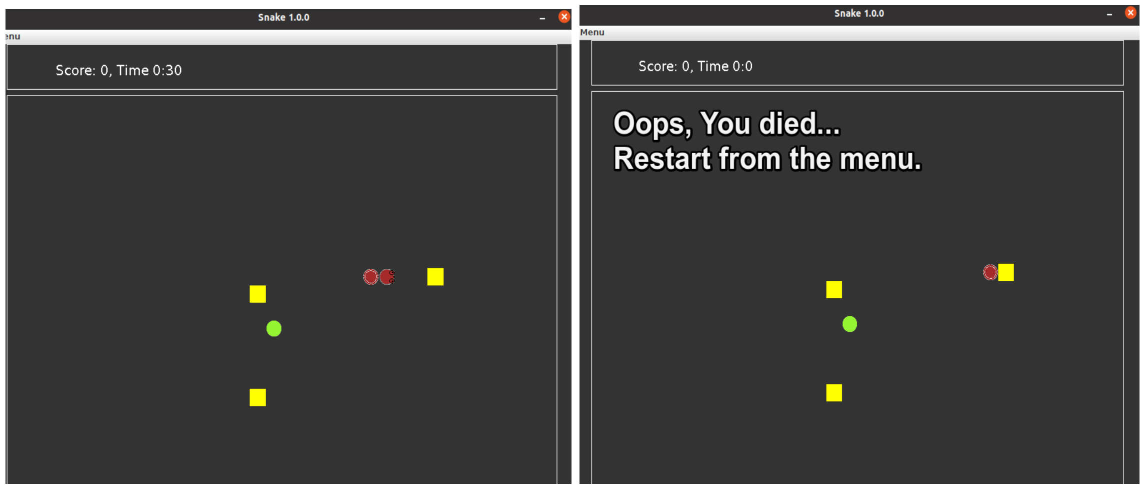
Score: 1, Time 0:40

fig 2.

3. ***Snake's length grows after eating food***

   fig 2.

4. ***Game finishes if snake collides with rock / Timer starts with the program and finishes when game is over.***



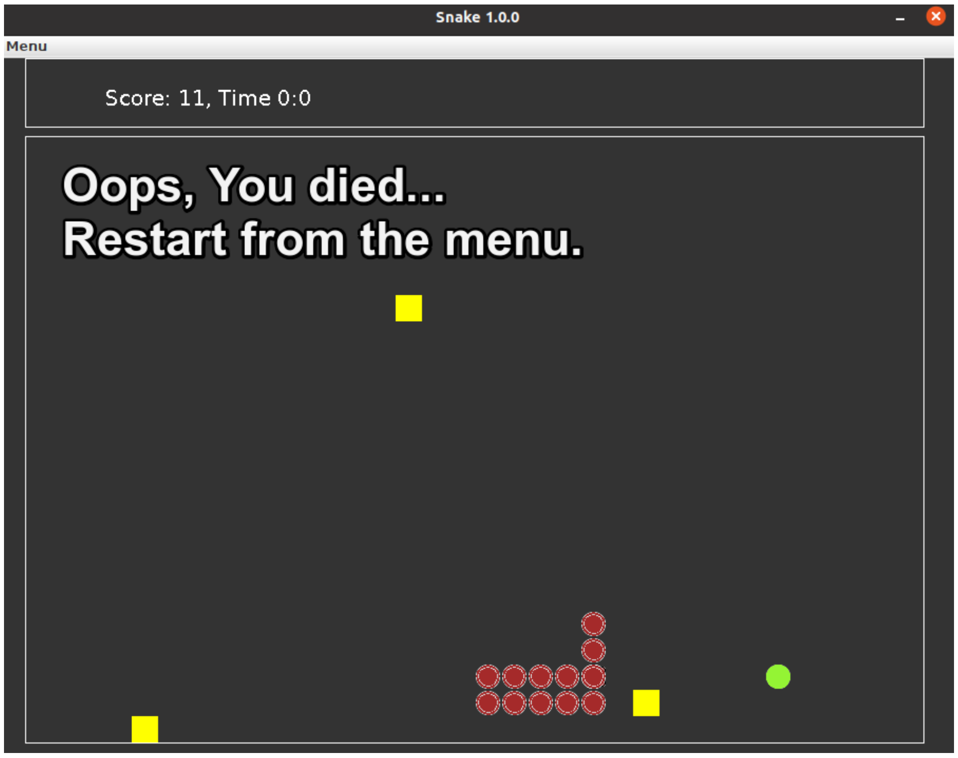
Score: 0, Time 0:30

Score: 0, Time 0:0

Oops, You died...
Restart from the menu.

fig 3. before the collision after the collision: game is over.

5. ***Game finishes if the snake collides with itself.***

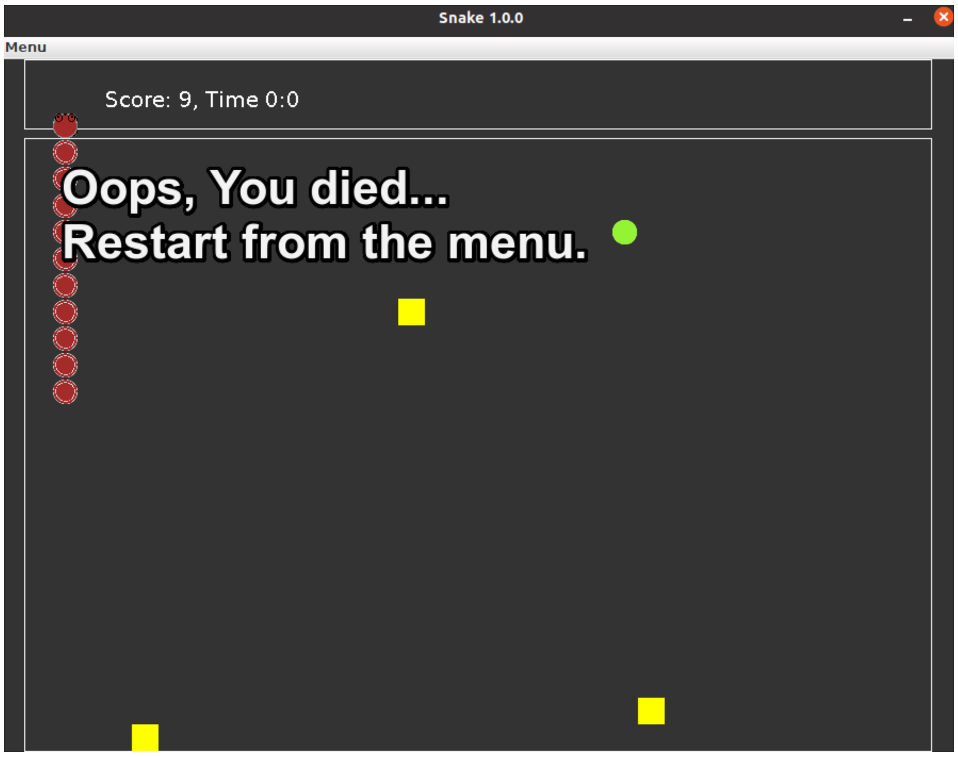


Score: 11, Time 0:0

Oops, You died...
Restart from the menu.

fig 4.

6. ***Game finishes if snake hits the wall***

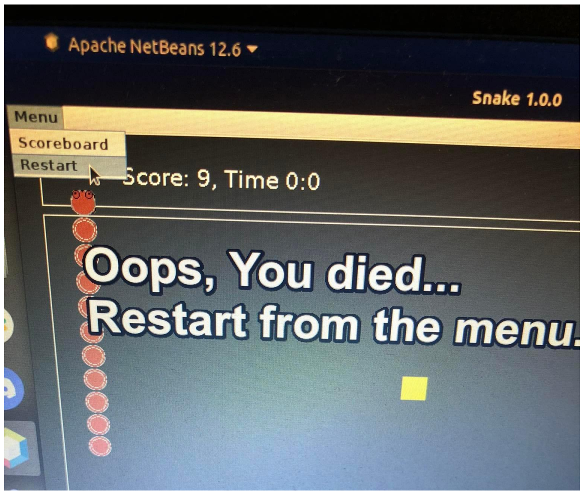


Score: 9, Time 0:0

Oops, You died...
Restart from the menu.

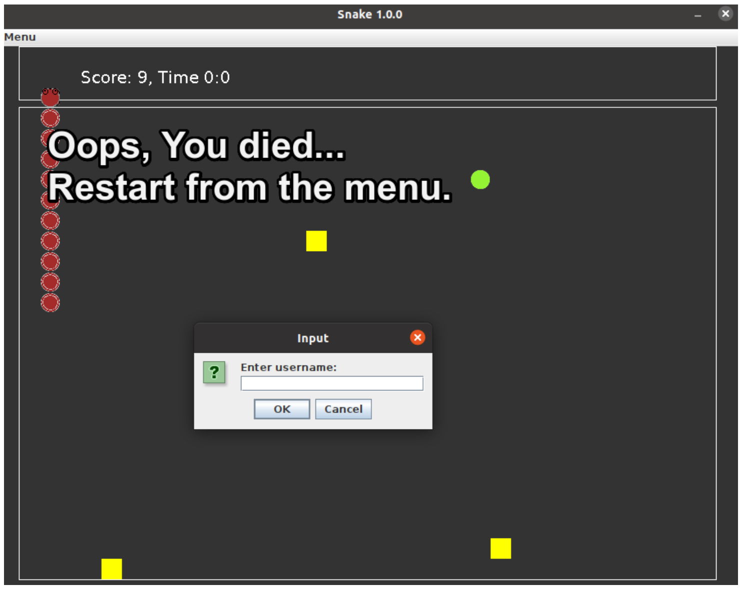
fig 5.

7. ***Restart menu***



Score: 9, Time 0:0

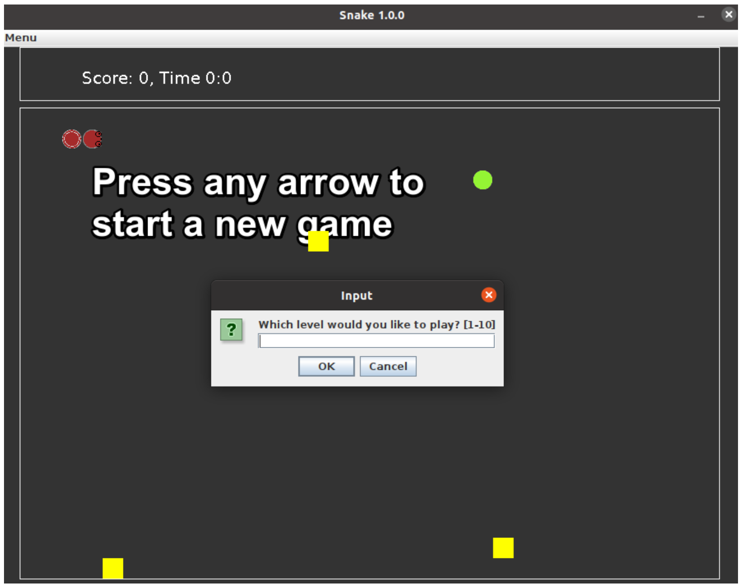
Oops, You died...
Restart from the menu.



Score: 9, Time 0:0

Oops, You died...
Restart from the menu.

Enter username:



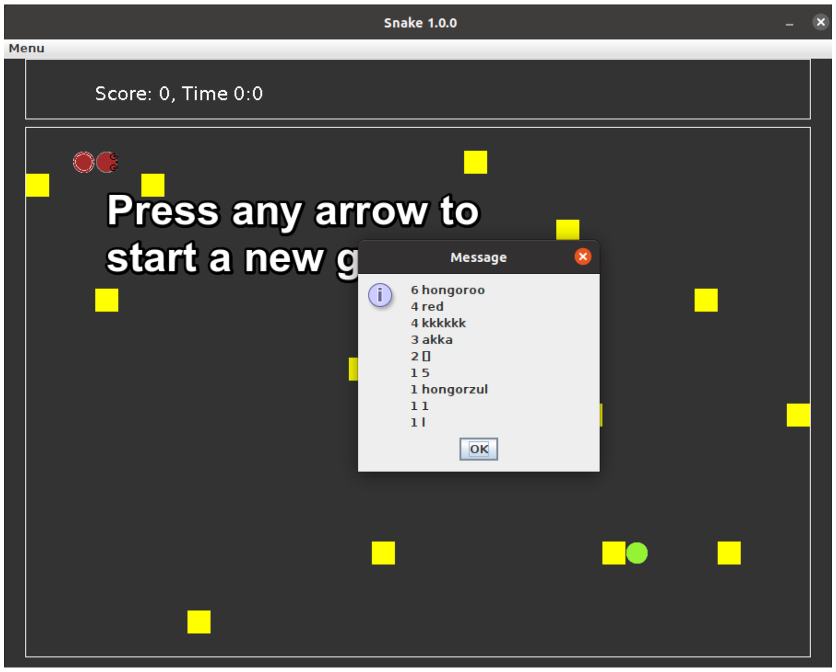
Score: 0, Time 0:0

Press any arrow to start a new game

Which level would you like to play? [1-10]

fig 6.

8. ***Top ten player scores menu***

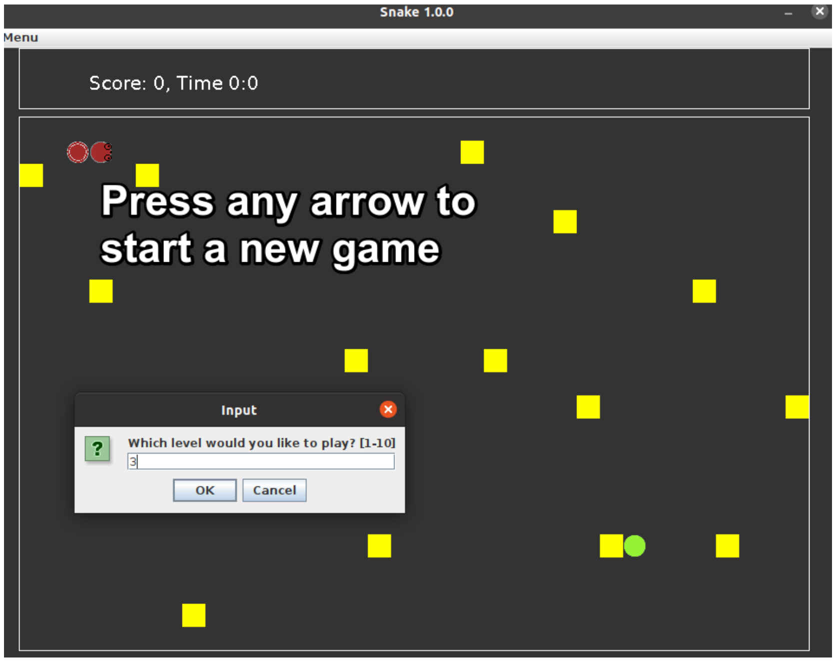


Score: 0, Time 0:0

```
6 hongoroo
4 red
4 kkkkkk
3 akka
2 []
1 5
1 hongorzul
1 1
1 l
```

fig 7.

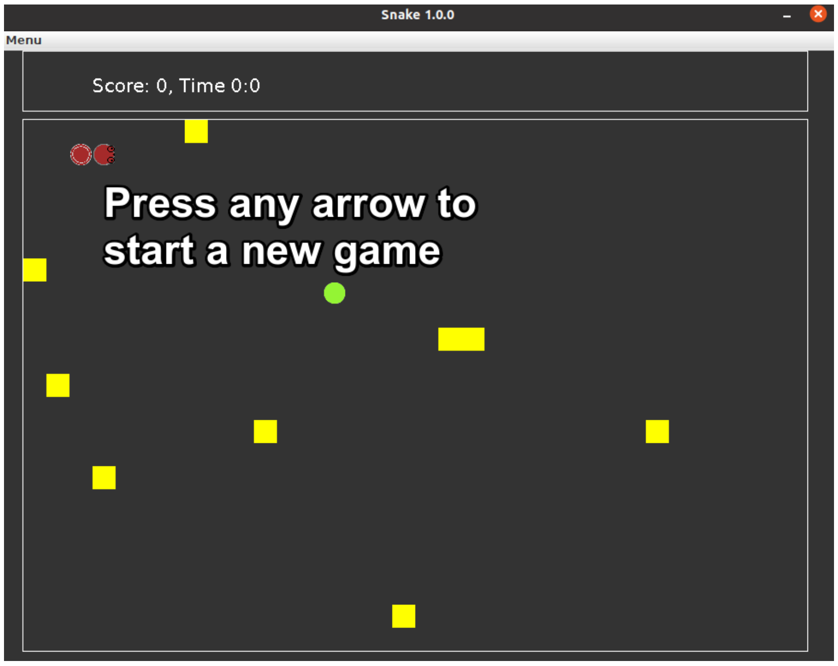
9. ***Difficulty level is based on the randomly located but increased barrier numbers.***
example: level 3 has 9 rocks located randomly



Score: 0, Time 0:0

Which level would you like to play? [1-10]

3

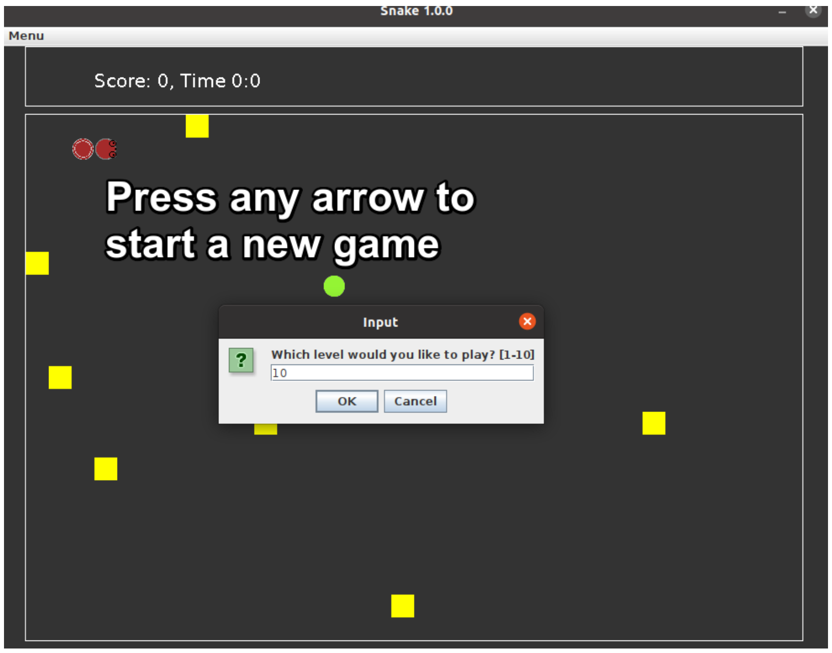


Score: 0, Time 0:0

Press any arrow to start a new game

fig 8.

example: level 10 has 30 rocks located randomly

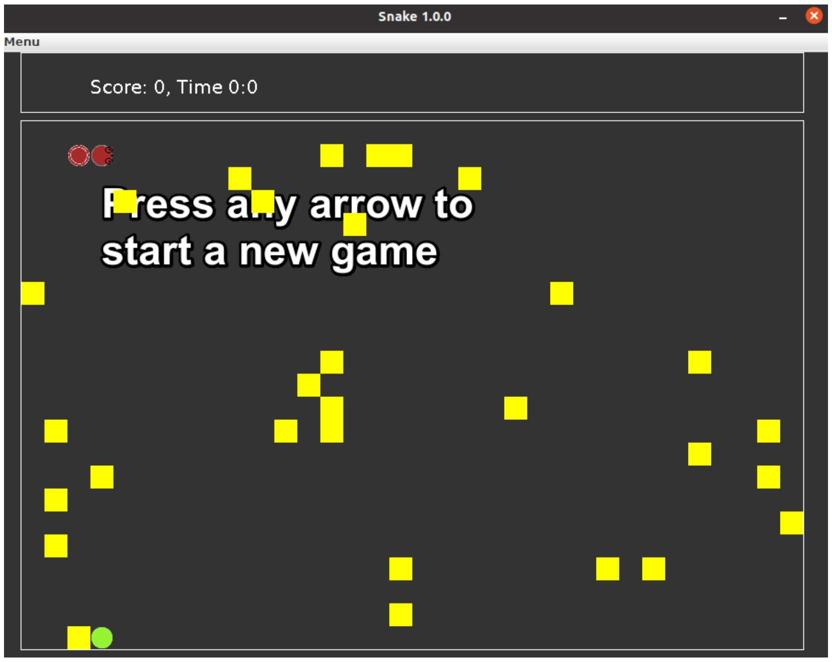


Score: 0, Time 0:0

Press any arrow to start a new game

Which level would you like to play? [1-10]

10



Score: 0, Time 0:0

Press any arrow to start a new game

fig 9.