# Snake Game Algorithm

**Major tasks:**

1. redraw the snake
2. check if any key is pressed
3. do something for each key pressed
4. move snake's head in the current direction of movement
5. move snake's body after the head

**Break the major tasks to smaller ones:**

**redraw the snake**

* + fill the window with black colour
  + redraw the head and all other segments with random colour

**do something for each key pressed**

* + if left arrow is pressed change the current direction of movement to left

(same for the other 3 directions)

**Keep breaking the tasks down:**

**redraw the head and all other segments**

* + loop through the list of segment's coordinates
  + pick a random colour
  + draw a segment with that colour

**Modifications (examples):**

Use different colours for the head and the body segments

* + redraw the head **with head's colour**
  + redraw all other segments **with segment's colour**

Prevent snake from going backwards into itself

* + if left arrow is pressed **and the snake is NOT moving to the right**,change the current direction of movement to left

(same for the other 3 directions)

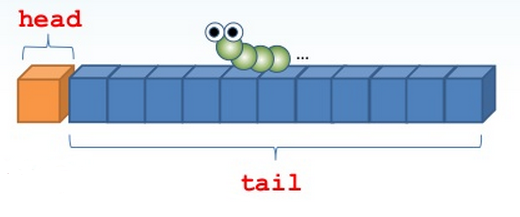
Detect when snake cuts into itself - game over

* **if sneak's head overlaps a segment from snake's body, exit and show game over screen**

Add an "apple" at random location

* **generate random appleX and appleY coordinates from a grid list**
* when redrawing the snake, **redraw the apple as well**

# Snake Game Assignment DUE DATE : Nov 20

Download **snake\_template.py** from Moodle and do the following:

MANDATORY features:

* Use different colours for the head and the body segments.
* Change segment size and spacing, so the snake segments are fully separated (no overlapping)
* **Limit snake's movement** within the window borders. If snake goes out - **game over or comes from the opposite side**
* **Prevent** snake from going backwards into itself.
* Detect when snake cuts into itself - **game over.**
* Add an "apple" at random grid location on the screen – so the snake head position will match the apple position horizontally and vertically
* Detect when the snake eats the apple.
* Make snake longer when an apple is eaten and remove the action of the space bar.
* Add apple generator - next apple appears on the screen after the current is eaten.
* Add a counter that counts how many apples are eaten. Print the score.
* Change snake's speed, after certain number of apples are eaten.
* Add a timer that counts down to zero. It resets back to max every time an apple is eaten otherwise game is over
* Add a background image, Add an intro screen and game over screen

Suggestions for EXTRA features:

* use different color for the new segments, while extending the snake and then convert to the regular color when the next segment is added ❸
* add sound to the game and special sound when an apple is eaten – see example ❹
* "boost" remaining time up for eating two apples in a very short period of time ❹
* generate poisonous “apples” (green or yellow) that will shrink the snake or reduce it to its original size ❹
* add more apples at a time regardless when the first one is eaten. ❹
* generate different size of "apples" – bigger apples lead to longer snake ❹
* change shape of snake's head and tail segments to triangles or add images for them❺
* Add obstacles (walls or saws) that will prevent the movement of the snake or kill the snake.❺

**Note: You don't have to implement all suggested features. Complete as many Extra features as needed for 12 marks. NO BONUS MARKES**

**Read carefully the rubric and submit it, with self-assessment filled.**

# Snake Game RUBRIC DUE DATE :Nov 20

**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Self-assess** | **marks out of** | **Teacher eval** |
| ***Application Total*** |  | ***25*** |  |
| segment size and spacing have been adjusted, so the snake segments don’t overlap |  | **1** |  |
| head and body segments **are different colours** |  | **2** |  |
| action of the space bar is removed and added to the collision check |  | **1** |  |
| if snake exits the screen: *a)* game is over or *b)* snake enters from the opposite side |  | **2** |  |
| game is over if snake cuts into itself |  | **3** |  |
| prevent snake from going backward into itself *(no game over)* |  | **3** |  |
| Apples (one by one) are generated at random location on the screen.  Full marks if apples are generated on a grid location - search random module for a function |  | **3** |  |
| snake "eats" an apple when it goes through it - the apple disappears |  | **3** |  |
| count/score is updated when the snake eats an apple |  | **2** |  |
| Timer starts at **20** seconds and counts down till the apple is eaten (reset to 20 every time the apple is eaten) If the count goes to zero – game is over |  | **5** |  |
| ***Thinking (fill in the extra features you have chosen to complete and the marks) Total*** |  | ***17*** |  |
| Extra Feature #1 |  |  |  |
| Extra Feature #2 |  |  |  |
| Extra Feature #3 ***12 points*** |  |  |  |
| **Code is efficient and unique(submit through Turn-It-In)**  **Students work independently during class time**  **No troubleshooting help required** ❷ |  | **5** |  |
| ***Communication Total*** |  | ***10*** |  |
| All variable names and function names are descriptive and start with lower case letters ( except the pre-set color and dimensions) |  | **2** |  |
| Code is well organized and formatted:   1. all Images and fonts are imported before the execution section. 2. all functions are organized into a separate section and labeled. 3. code in all functions is sorted and organized into sections ( render, blit) 4. functions are less than a string long – each completes specific task |  | **4** |  |
| All variables are declared, organized into sections and **commented at the beginning** of the program (*snake attributes*) |  | **2** |  |
| Clear and understandable comments that elaborate and explain the logic of the function/structure/command |  | **2** |  |
| ***Knowledge Total*** |  | ***12*** |  |
| The game changes speed without stretching the segments. No lagging |  | **3** |  |
| The game has a background image instead of just black screen. |  | **1** |  |
| The game has a counter – showing how many apples are eaten. |  | **3** |  |
| The game has separate intro and game over functions are created. They are controlled correctly with while loops before and after the main. – use Booleans |  | **4** |  |
| The self-assessment rubric are completed correctly – not all 100% |  | **1** |  |

**Comments:**