Group # 39

Snake++

Wentao Lin
Nader Hosseini Naghavi
Piranavan Jeyakumar
Tianci Du
Ziqingqing Ye

Project Description ~ Based on the classic game, Snake, the user will be controlling a snake using the keyboard (e.g Up, Left, Down, Right). Optionally, the user will be able to control the snake using hand gestures, where a specific hand gesture will control one aspect of the snake's movement. Additionally, "fruits" and "bombs" will be randomly generated on the screen, the fruits serve to increase the player's score, but also the size of their snake, whereas if the snake collides with a bomb, the game will be terminated. As the game progresses, more bombs and less fruits will be randomly generated on the screen (to make the game more challenging)! The goal of the game is to get the highest possible score, without colliding with the bombs or itself.

Features ~

Required:

- The game should enable the snake character to move up, go down, turn left or turn right according to the instructions of the user provided by keyboard.
- The fruit, which can be eaten by the snake to increase the player's score, will be displayed randomly in the graphical interface.
- The bombs, which when collided with, will result in the end of the game, can be displayed randomly in the graphical interface.
- The length of the snake will become longer after a fruit is eaten, resulting in the increase of the player's score.
- A certain number of fruits must be displayed on the screen at all times, once a fruit is eaten, a new one must be randomly generated elsewhere on the graphical interface.
- After colliding with itself or the bombs, there would be a collision effect shown on the screen and result in the termination of the game.
- The game will record the score the player earns each time and display the top 5 scores in decreasing order upon the termination of the game. The player's current score will be displayed whilst playing.
- When the snake moves off one side of the screen, it will continue moving from the opposite side of the screen.
- As the game progresses (based on time or score), more bombs will be randomly generated on the screen (max of 10 bombs) and less fruits will be randomly generated (min of 1 fruit).

Optional:

- Instead of using the keyboard/mouse as the controller to move the snake, this game enables hand gestures/hand movements as the controller. The hand gestures/hand movements can be captured using the webcam (using the OpenCV library) so that the snake can move according to their instructions.
- Replacing the bombs with block obstacles of VARYING SIZES which will be randomly generated on the graphical interface. Being sure to

generate the obstacle a fair distance away from the snake, to add a layer of challenge, but also being fair enough so that the snake can work around it.

- Based on the preference of users, they might be able to change the color of the background or the color of the snake.
- Background music might be played as players are playing the game.

Wish-List:

- Develop a 2-player game mode in addition to the single player game mode. In the 2-player game mode, 2 snakes will be moving at the same time, more fruits and less bombs will be generated, but both snakes CANNOT collide with: each other, any bombs, and themselves.
- By supporting Bluetooth connection, multiple users can connect multiple wireless controllers to control the different snakes on the screen.
- In addition to gestures or hand movements control, controlling the snake via brain waves may be used.

Risks ~

Risks related to using OpenCV for capturing hand movements:

- One possible risk could be establishing a stable connection between OpenCV and the computer's webcam for capturing hand movements.
  We can try to use a stable library of OpenCV for making a stable connection with the webcam in our C++ code.
- The other risk related to OpenCV, could be the delay between capturing the hand gestures from the webcam and feeding instructions to the snake game. The delay may appear when: OpenCV recognizes hand gestures (based on the processor running the object detection algorithm), encoding the hand gestures as movement commands, and feeding the movement commands to the main game's code. Based on the processing power of the computer running the algorithm, these steps could be time consuming causing a lag or delay in the main game. This could be an issue since in the snake game, the user's commands should be executed quickly; otherwise, the game will terminate very soon.
  To solve this issue, we could use the Cuda package to run the object detection algorithm of OpenCV on GPU rather than CPU.

Risks related to the main Snake game:

- It may be difficult to implement the termination rule of the snake. In the original game, we would use an algorithm to detect the collision of the snake with itself or the bombs as a signal for termination of the game and in the optional features we should also identify the collision of the snake with obstacles (of varying sizes). We can use graphical libraries of the C++ and/or conditional statements to detect the termination events.
- Generating fruits on random locations on the screen could be problematic. We must ensure the location of the fruits are random, but also ensure that they are not

generated within the snake OR too close to the snake's head (otherwise collecting fruits would be too easy).

- In terms of obstacles/bombs, one risk could be generating them on random locations like the fruits (if a bomb/obstacle is generated too close to the head, this may end in an unfair termination of the game). The other risk could be determining the right shape for them so that the player can easily distinguish between them and the fruits. We can use a certain shape or even a distinct color for the obstacles to make them different from the fruits.
  The other risk related to obstacles, would be generating them with different sizes. We can use an algorithm to increase the size of obstacles as the player progresses in the game.
- One possible risk would be reading the player's commands from the keyboard/mouse. We can use C++ input utilities to read the keyboard/mouse commands.

Issue with the graphical output of the game:

- One challenge of the game would be showing the game accurately in a graphical interface. We can use C++ graphical libraries for making the graphical interface of the game.
- If we only use C++ for making the graphical interface of the game, image rendering might not be done properly and the image of the snake may not be able to keep up with it's movement.

Other Notes ~

In this project, we will use a desktop computer system to run our C++ software of the original game and its additional features (like hand gesture detector). We will use I/O utilities of the C++ programming language to make a graphical interface for our snake game and receive the commands from the player. In addition, we will use some computational algorithms in C++ to generate fruits and obstacles/bombs in random locations and with different shapes. We would also define some rules for the interaction of the main snake with these objects.

In terms of the optional features of our program, we will use the C++ version of the OpenCV library which is a library mainly used for Machine Vision applications. We will use the Object Tracking feature of this library to detect the hand movements of the player. For our project, we will need basic C++ knowledge and skills for developing the required computational algorithms. We also need knowledge of C++ graphical utilities and libraries that we plan on studying these libraries and learning how to use them. For the optional part of our project, we will need the OpenCV library for hand gesture and movement detection. One of our members has previously worked on a project related to object tracking in C++ using OpenCV that could be very useful for our project.