| Topic | SNAKE GAME - FINAL | |
|---|---|---|
| **Class Description** | **Student will complete the snake game. They will incorporate winning and losing conditions to the game. Additionally, they will write the code to reset the game once the game is over.** | |
| **Class** | **PRO C272** | |
| **Class time** | **45 mins** | |
| **Goal** | • Learn to use game states.<br>• Incorporate winning and losing conditions.<br>• Write code to show winning and losing messages..<br>• Learn to write code to reset the game using the joystick's push button. | |
| **Resources Required** | • Teacher Resources:<br>  ○ Laptop with internet connectivity<br>  ○ Earphones with mic<br>  ○ Notebook and pen<br>  ○ Smartphone<br>• Student Resources:<br>  ○ Laptop with internet connectivity<br>  ○ Earphones with mic<br>  ○ Notebook and pen | |
| **Class structure** | **Warm-Up**<br>**Teacher-Led Activity**<br>**Student-Led Activity**<br>**Wrap-Up** | **10 mins**<br>**15 mins**<br>**15 mins**<br>**05 mins** |

| WARM-UP SESSION - 10 mins | |
|---|---|
| **Teacher Action** | **Student Action** |
| Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? | **ESR**: Hi, thanks!<br>Yes, I am excited about it! |

| | |
|---|---|
| **Following are the WARM-UP session deliverables:**<br>• Greet the student.<br>• Revision of previous class activities.<br>• Quizzes. | Click on the slide show tab and present the slides |
| **WARM-UP QUIZ**<br>Click on In-Class Quiz | |
| **Activity Details**<br><br>**Following are the session deliverables:**<br>• Appreciate the student.<br>• Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. | |
| **TEACHER-LED ACTIVITY - 15mins** | |
| **Teacher Initiates Screen Share** | |
| • **Writing code to show a welcome message to the player.** | |
| **Teacher Action** | **Student Action** |
| Do you remember what we did in the previous class?<br><br>Do you have any doubts?<br><br>*If the student has any doubts, clarify the doubts.* | **ESR**: We had started with the snake game. We had implemented the snake's movement and resolved the bugs.<br><br>**ESR:** Varied. |

| | |
|---|---|
| In the class, we will complete the **Snake Game.**<br><br>Let's discuss the things we want to achieve today.<br><br>What do we want to do next? | |
| | **ESR:** We need to add the winning and losing condition to this game |
| Great! Let's discuss the winning and losing conditions too.<br><br>What can be the winning condition? | |
| | **ESR:** When the snake eats a certain number of food or it has reached a certain length, it will win the game. |
| What should be the losing condition? | |
| | **ESR:** When it touches its tail by mistake while moving. |
| Great!<br><br>Is anything else left? | |
| | **ESR:** Varied. |
| We didn't add any messages for the player yet. We need to add a welcome message to the game. We also need to add messages when the player wins or loses.<br><br>We know our goals for today's class now. Ready to start the project? | |

Let's open the previous class code.

*The teacher downloads the boilerplate code and uploads these files to the wokwi simulator.*

Let's add some welcome text for the player's first.

But before the welcome text comes up on the screen, we want to alert the user. For that, we will write a function which will switch on and switch off all the LEDs thrice to grab the player's attention.

1. First, we will define a function named **all_animate()**. This function will switch on all the LEDs for 500 milliseconds and switch off the LEDs for 500. It will do this thrice to alert the user.
    - Define a new function named **all_animate()**
    - We will use the **setPoint()** function to light up the LEDs.
    - To light up all the LEDs, we will use a for loop()

```
for (int x = 0; x < 8; x++){
    for (int y = 0; y < 8; y++){
      matrix.setPoint(y,x,true);
    }
  }
```

    - Now, let's keep these lit up for 500 milliseconds.

```
delay(500);
```

- Then, we switch off the LEDs for 500 milliseconds.

```
matrix.clear();
delay(500);
```

- Write a **while()** loop around these lines of code to make all the LEDs light up thrice.

Your code should look like this-

```
void all_animate(){
  int i=0;
  while(i<3){

    for (int x = 0; x < 8; x++){
      for (int y = 0; y < 8; y++){
        matrix.setPoint(y,x,true);
      }
    }
    delay(500);
    matrix.clear();
    delay(500);
    i++;
  }
}
```

Now, let's write the code to show some welcome text. We will write a generalized method named **show_message()** so that we can reuse this function to show other messages too.

- Define **show_message()** method.
- **show_message()** method should have one parameter which should be a character array or a string. We will pass the message to the method through this parameter.

```
void show_message(char messageToShow[]){



}
```

- We will call the **all_animate()** method in the **show_message()** method.

```
void show_message(char messageToShow[]){

  all_animate();

}
```

- Then we can use the **setChar()** method to print each letter of the message passed through the method. Each letter should be shown for 500 milliseconds.

  Let's say, we want to print "W", we can write-

```
matrix.setChar(6 , 'W');
delay(500);
```

- But our message is stored as a string. How can we access each character of the array?

**ESR:** We can use the index of the array to access each character. We can do it using a loop.

| | |
|---|---|
| Great! But when should this loop end?<br><br>Exactly! In c, every string ends with **'\0'**.<br><br>**'\0'** is called the null character. This null character helps us to understand where the string ends and it is added automatically at the end of the string.<br><br>● Initiate an integer named i to 0. This variable will help us hold the index of each character of the string.<br><br>`int i=0;`<br><br>● Add a **while** loop. This **while** loop should run until the last character of the string is accessed. So, the condition should be **messageToShow[i] != '\0'.** | **ESR:** After we have printed the last character. |

```
void show_message(char messageToShow[]){

  all_animate();

  int i=0;

  while(messageToShow[i]!='\0'){

  }

}
```

- Inside the loop, use **setChar()** code to show each character of the string for 500 milliseconds.

```
matrix.setChar(6 , messageToShow[i]);
delay(500);
```

- Use **clear()** method to clear up the display for 500 milliseconds.

```
matrix.clear();
delay(500);
```

- Lastly, increase the index variable i by one so that we can access the next character.

```
i++;
```

Your code should look like this-

```
void show_message(char messageToShow[]){

  all_animate();

  int i=0;

  while(messageToShow[i]!='\0'){
    matrix.setChar(6 , messageToShow[i]);
    delay(500);
    matrix.clear();
    delay(500);
    i++;
  }

}
```

| | |
|---|---|
| Call the **show_message()** method in the **setup()** method. Pass “WELCOME” as the argument. | |

```
void setup(){

  Serial.begin(9600);

  matrix.begin();
  matrix.clear();

  //  setting debounce time
  button.setDebounceTime(25);

  //  welcome text
  show_message("WELCOME");
}
```

| | |
|---|---|
| Now, we have to write the code for winning and losing conditions. I want you to find a solution for these two tasks. | |
| Teacher Stops Screen Share | |
| Can you solve it?<br><br>Let's try. I will guide you through it. | **ESR:** Yes, sure! |
| **STUDENT-LED ACTIVITY- 15 mins** | |
| • **Ask the student to press the ESC key to come back to the panel.**<br>• **Guide the student to start Screen Share.**<br>• **The teacher gets into Full Screen.** | |
| **Student Initiates Screen Share** | |

<table>
<tr><td colspan="2">ACTIVITY<br><br>• Add game states.<br>• Write code for winning and losing conditions.<br>• Add functionalities to restart the game.</td></tr>
<tr><td>Teacher Action</td><td>Student Action</td></tr>
<tr><td>Teacher guides the student to download the boilerplate code from Student Activity 1.</td><td>Student downloads the boilerplate code.<br>Student opens wokwi simulator and upload the boilerplate code.</td></tr>
<tr><td>In our initial classes, we learnt about game states. Do you remember what game states are?<br><br>Exactly. It will help decrease complexity of our code when we add the winning and losing conditions.<br><br>We will have three gameStates in our game.<br><br>1. Play, which we will indicate with 0<br>2. Win, which we will indicate with 1<br>3. Lose, which we will indicate with 2<br><br>Let’s define a variable named gameState at the top. We</td><td>ESR: Yes. Game state helps us understand which state our game is in. The game state can be stored in a variable. With the help of the game state variable, we can tell the computer which section of code to run in which state.</td></tr>
</table>

will initiate it at 0.

```
int gameState=0;
```

Now, let's look at the **loop()** method now and incorporate the gameState there.

The code that we have written till now in the loop method should only run in the **play** state.

So, let's put this section of code in an **if** condition.

```
void loop(){

  //  start noting down the time
  if(gameState==0){
    current_time = millis();
    check_direction();  //  check direction of joystick rotation
    if (current_time - prev_time  >=  threshold){
      prev_time = current_time;
      matrix.clear();  //  clear display before every frame
      move_sprite();  //  move the sprites
      update_snake();  //  updating snake array
      draw_sprite();  //  draw sprites
    }
    food_eat_check();  //  checking if snake has eaten food

  }
  //  for better working of simulator and controlling snake speed
  delay(10);

}
```

| | |
|---|---|
| Great! Let's implement the winning condition .<br><br>What should it be?<br><br>Where should we write this section of code?<br><br>Correct! We should write the code for the winning condition inside the **food_eat_check()** method. | **ESR:** When the player eats 15 food objects, the player wins the game.<br><br>**ESR:** We should check this when the snake eats the food. This is done by the **food_eat_check()** method. So, we should write this section of code inside the **food_eat_check()** method. |

```
void food_eat_check(){

  //  check if head collides with food
  if (head_x  ==  food_x  &&  head_y  ==  food_y){

    //  change the food coordiantes
    food_x = random(0,8);
    food_y = random(0,8);

    //  changing snake_length
    snake_length++;
    if (snake_length >= 15){



    }

    //  decrease the threshold
    threshold = threshold - 50;
    if (threshold < 100)threshold = 80;
  }
}
```

What do we want to do when the player wins?

Great. Let's write that code.

**ESR:** We want to change the **gameState** to 1.

```
void food_eat_check(){

  //  check if head collides with food
  if (head_x  ==  food_x  &&  head_y  ==  food_y){

    //  change the food coordiantes
    food_x = random(0,8);
    food_y = random(0,8);

    //  changing snake_length
    snake_length++;
    if (snake_length >= 3){
      gameState=1;
    }

    //  decrease the threshold
    threshold = threshold - 50;
    if (threshold < 100)threshold = 80;
  }
}
```

| What do you want to do when the gameState is 1? | **ESR:** Let's show a message saying "YOU WON". |
|---|---|

```
void loop(){

  //  start noting down the time
  if(gameState==0){
    current_time = millis();
    check_direction();  //  check direction of joystick rotation
    if (current_time - prev_time  >=  threshold){
      prev_time = current_time;
      matrix.clear();  //  clear display before every frame
      move_sprite();  //  move the sprites
      update_snake();  //  updating snake array
      draw_sprite();  //  draw sprites
    }
    food_eat_check();  //  checking if snake has eaten food

  }

  if(gameState==1){
    show_message("YOU WON");
  }

  //  for better working of simulator and controlling snake speed
  delay(10);

}
```

| | |
|---|---|
| Now, what's next?<br><br>We want the game to end when the snake's head touches its own body. To do that we will define a new method called **snake_collision_check().** Let's start.<br><br>● Define the **snake_collision_check()** method. | **ESR:** let's add the losing condition. |

- Snake's head's index is stored at the 0th index of **snake_x** and **snake_y**. We want to compare the head's position with the rest of the body's position i.e. each dot in the snake's body except the head.

  If the position of the head and one of the dots from the snake's rest of the body has same x,y value, then we can say that the snake has collided.

  We will write this code using a **for** loop which will loop through the 1st to last index of the snake's body. We will compare each of these positions with the 0th index.

```
void snake_collision_check(){

  //  checking if head collides with any other body part
  for (int i = 1; i <= snake_length; i++){
    if (snake_x[0]  ==  snake_x[i]  &&  snake_y[0]  ==  snake_y[i]){
      gameState=2;
    }
  }

}
```

Now, what do we want to do when gameState is 2 or the player has lost?

**ESR:** Let's show a message again.

```
void loop(){

  //  start noting down the time
  if(gameState==0){
    current_time = millis();
    check_direction();  //  check direction of joystick rotation
    if (current_time - prev_time  >=  threshold){
      prev_time = current_time;
      matrix.clear();  //  clear display before every frame
      move_sprite();  //  move the sprites
      update_snake();  //  updating snake array
      draw_sprite();  //  draw sprites
    }
    food_eat_check();  //  checking if snake has eaten food

  }

  if(gameState==1){
    show_message("YOU WON");
  }

  if(gameState==2){
    show_message("GAME OVER");
  }

  //  for better working of simulator and controlling snake speed
  delay(10);

}
```

| | |
|---|---|
| Call the **snake_collision_check()** method in the **play** state. | |

```
if(gameState==0){
  current_time = millis();
  check_direction();  //  check direction of joystick rotation
  if (current_time - prev_time  >=  threshold){
    prev_time = current_time;
    matrix.clear();  //  clear display before every frame
    move_sprite();  //  move the sprites
    update_snake();  //  updating snake array
    draw_sprite();  //  draw sprites
  }
  food_eat_check();  //  checking if snake has eaten food
  snake_collision_check();
}
```

Great work. One last thing left though. Can you guess what it is?

Exactly! Once the game is over, we want to add a reset option for the game.

1. Write another **if** condition and pass **gameState** as the condition. This condition will work when gameState is not 0 i.e. it will work in both **Win** and **Lose** state. We will write the code at the of **loop()** method so that it works after the “GAME OVER” message is displayed.

```
if (gameState){

}
```

**ESR:** We might want to play the game again and reset it.

2. We will now write an infinite loop inside this **if** statement. This is because we want it to wait here until the game restarts.

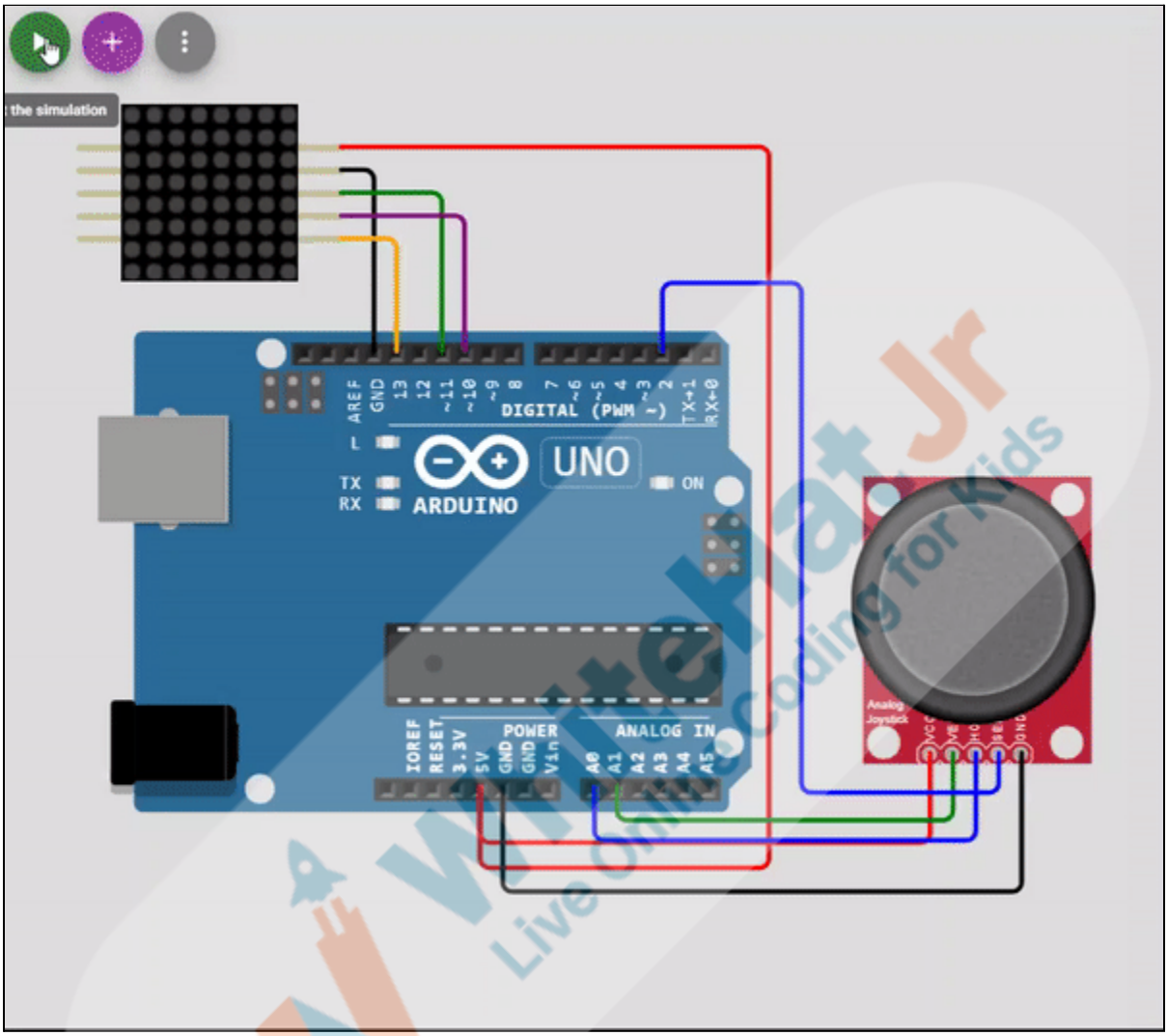
```
if (gameState){
  while(true){

  }
}
```

3. We want to restart the game when the joystick's push button is pressed. Call the **button.loop()** function here.
4. Now, if the button is pressed, reset the **gameState** to 0, so the game is in **play** state again. Also, reset all the other variables.

```
if (button.isPressed()){

  //  resetting variables
  head_x = 0, head_y = 0, snake_length = 0, threshold = 800;
  direction = "" , prev_direction = "";
  for (int i = 0; i < 64; i++){
    snake_x[i] = 0;
    snake_y[i] = 0;
  }
  gameState = 0;
  break;
}
```

```
if (gameState){

  while(true){
    button.loop();
    if (button.isPressed()){

      //  resetting variables
      head_x = 0, head_y = 0, snake_length = 0, threshold = 800;
      direction = "" , prev_direction = "";
      for (int i = 0; i < 64; i++){
        snake_x[i] = 0;
        snake_y[i] = 0;
      }
      gameState = 0;
      break;
    }
  }
}
```

**Reference Output:**




Click here to view the full output video.

| | |
|---|---|
| You have done a great job today! | |

**Teacher Guides Student to Stop Screen Share**

**WRAP-UP SESSION - 05 mins**

**Activity details**

**Following are the WRAP-UP session deliverables:**

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

**WRAP-UP QUIZ**
Click on In-Class Quiz

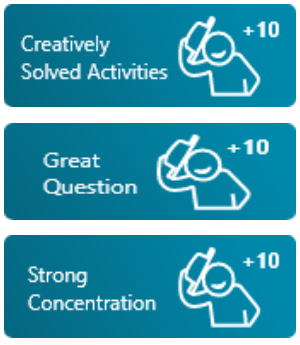
**Activity Details**

**Following are the session deliverables:**

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

**FEEDBACK**

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

| Teacher Action | Student Action |
|---|---|
| You get "hats-off" for your excellent work!<br><br>In the next class, we will start building a digital reading device. | *Make sure you have given at least 2 hats-off during the class for:*<br><br>  |

**PROJECT OVERVIEW DISCUSSION**

| Refer the document below in Activity Links Sections | |
|---|---|
| **Teacher Clicks** ✖ End Class | |
| **ADDITIONAL ACTIVITIES** (Optional) | |
| **Additional Activities** | |

| ACTIVITY LINKS | | |
|---|---|---|
| **Activity Name** | **Description** | **Links** |
| Teacher Activity 1 | Simulator | https://wokwi.com/ |
| Teacher Activity 2 | Previous Class Code | https://github.com/procodingclass/PRO-C271-Reference-Code |
| Teacher Reference 1 | Teacher Reference Code | https://github.com/procodingclass/PRO-C272-Reference-Code |
| Teacher Reference 2 | Project | https://s3-whjr-curriculum-uploads.whjr.online/51f6b439-96b5-4db7-9b10-857fe19ffc11.pdf |
| Teacher Reference 3 | Project Solution | https://github.com/procodingclass/PRO-C272-Project-Solution |
| Teacher Reference 4 | In-Class Quiz | https://s3-whjr-curriculum-uploads.whjr.online/c350c191-f801-4f06-8259-f0779b718b9b.pdf |
| Student Activity 1 | Boilerplate code for Student Activity | https://github.com/procodingclass/PRO-C272-Student-Boilerplate |