PRO-C19

CHECKPOINT REVISION CLASS: INFINITE RUNNER GAME

What is our GOAL for this MODULE?

In the past 10 classes, you have learned many new concepts and created an infinite runner game—Trex. Today was a capstone class to revise and rewrite all the concepts learned in the last few classes.

What did we ACHIEVE in the class TODAY?

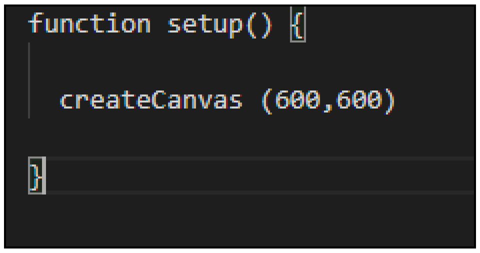
- We practiced and implemented different programming constructs to strengthen concepts used in building an Infinite Runner Game.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Creating infinite space - to create continuous scrolling ground/background.
- Adding Images and Game Sounds - to make games attractive and more fun.
- Spawning random objects - spawn different obstacles or rewards in the game.
- Memory Leak - to save memory/space in the RAM by clearing the unused memory.
- Collider & Collisions - each sprite has a collider around it which helps in detecting collisions.
- GameStates - to define the behavior of each object used in the game at different stages.

How did we DO the activities?

1. Start decomposing the task into smaller tasks.
2. Create a canvas of the desired size as per the requirement of the game in the function **setup().**

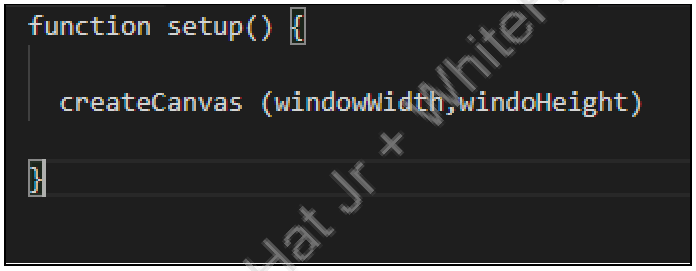


```
function setup() {

  createCanvas (600,600)

}
```

3. For the canvas size to be the same as device size use **windowWidth, windowHeight.**



```
function setup() {

  createCanvas (windowWidth,windoHeight)

}
```

4. Declare required Global variables which can be used across the program.
5. Load images, animation or sound files using function **preload().**
6. Create required sprites in function **setup().**
7. Add images/ animation to the sprites. (if needed)
8. Create a logic of the game in function **draw().**
9. Make it a practice of using separate functions and calling them inside function **draw()** as we did in **Trex** for **spawnClouds(), spawnObstacles(), and reset().**
10. If the game involves creating multiple sprites, remember to use **.lifetime()** to avoid **memory leak,** that is to save memory/space in the RAM by clearing the unused memory.
11. Adjust the sprite **collider** as per the requirement to identify **collision** between two objects.
12. Introduce **GameStates** to define behaviors of different objects at different stages of

the game.
13. To make the game mobile-friendly include **Touches.**
14. Once the game is complete, run the code to see the desired output.
15. Upload the code into **GitHub**.
16. **GitHub** link can also be used to generate a **.apk** file using **Thunkable.**

What's next?

In the next class, you will also be introduced to the concepts of Physics Engine.

Extend Your Knowledge:

1. Learn more about Touches by Mozilla Contributors (licensed under CC-BY-SA 2.5): https://developer.mozilla.org/en-US/docs/Web/API/TouchEvent/touches