Petan Ioana
Nàdia Pelegay Royo

Dino Game

GAMELOGIC

Our project is a video game in which the user is allowed to control and personalize its own dino. The purpose of this game is to jump over as many trees as possible. Every dino has 3 lives (stars) and the user can increase the difficulty of the game every time he loses a life.

Html

The html's structure is based on this key elements:

- The canvas: for setting the background of the game;
- The game status display: the div elements .gameStatus, .score, .gameOver, which keep track of the game progress and result;
- The personalization part: buttons and input fields for personalizing the character, including name input and color selection.
- Game Controls: Buttons to start, restart, and increase game difficulty.

Java script

1. `dino.js` - Dinosaur Movement

- `pressDownAction(event)`: Starts the jump when the space key is pressed.
- `jump()`: Animates the dinosaur's jump.
- `fall(jumpStartTime)`: Animates the dinosaur's fall back to the ground.

2. `personalise.js` - Game Personalization

- `personaliseGame()`: Initiates game personalization.
- `nameCharacter()`: Allows the player to choose a name.
- `textModified(event)`: Updates the displayed name.
- `colorCharacter()`: Allows the player to choose a color.
- `clickOnColor(event)`: Sets the dinosaur's color.
- `songChoose()`: Allows the player to choose a background song.
- `clickOnSong(event)`: Sets the selected song.

- `playChosenSong()`: Plays the selected song.
- `stopAllSongs()`: Stops all currently playing songs. ● `doneStep()`: Finalizes the personalization process.

3. `game.js` - Main Game Logic

- `startGame()`: Starts the game and schedules the first obstacle.
- `scheduleNextTree()`: Schedules the next tree obstacle.
- `endGame()`: Ends the game and displays the final score. ● `restartGame()`: Restarts the game after a collision.
- `restart2Game()`: Reloads the page to restart the game.
- `createNewTree()`: Creates a new tree obstacle.
- `moveTree(tree)`: Animates the tree's movement and checks for collisions.
- `checkCollision(tree)`: Checks for collisions between the dinosaur and a tree.
- `decreaseStars()`: Reduces the player's stars upon collision.
- `updateScore()`: Updates the score display.

CSS

- Purpose: Styles game elements like the dinosaur, buttons, stars, and game layout.
- Details: Defines the appearance and positioning of game elements for a cohesive design.

CHALLENGES& REFLECTION

Ioana

One of the hardest things I did in this project were the animations. Understanding, and most of all managing to make them all happen at the same time was really difficult. Making the trees slide at different intervals and then stopping each interval and animation as well as removing all the existing trees at the moment of collision was quite confusing. The jump of the dino was a really interesting part as I tried making it independent of pixels.

The elements that helped me the most were the functions as they made everything more organised even though there are many lines of code. I tried making the project more compact by sectioning it into different java script files. Trying to work with all of them at once was difficult.

Petan Ioana

Nàdia Pelegay Royo

Over all, working on this project became easier with time and I managed to solve the small mistakes that appeared, faster. Working with Nàdia was very helpful and enjoyable and made the project much more interactive and appealing. We tried combining our ideas and, in the end, the project became what we wanted. Together we made a simple game involving a dino that jumps over trees into an interactive musical, colourful and fun game. I am very happy with what we managed to do!

Nàdia

Personally, one of the first things I found challenging was organisation. Compared to the small and compact exercises we tackled in class, this project was massive; working with numerous JavaScript files, many constants, functions… it was a lot to manage. For this reason, I realised that not only was it important to name the constants and functions logically, in order to enhance readability and ease their retrieval posteriorly; but also to simplify and optimise the code to minimise repetition and reduce the concepts to keep track of, with classes like *'button'*.

Along the same lines, I think it was invaluable practice to work with a partner because you constantly have to collaborate, understand and combine codes. I have found it very interesting to understand Ioana's logic, to learn from it and to eventually apply it myself too. Additionally, discovering new functions has also been very enriching. In my case, I believe that the most out of the ordinary and complex function has been the whole implementation of the music. At first, I didn't even know if it was possible, but then working through and comprehending the differences in syntax and practice to apply it until success was indeed compelling.

This makes me think of the creativity and spontaneity that this project has involved. With Ioana, we went step by step developing ideas as they came. It has been marvelling to see how mere thoughts have materialised into an awesome, amusing and real game! Overall, I am very proud and satisfied with our job and I have adored getting to work with Ioana; it's been super fun!