p6game_new: Turn-based JS board game

Javascript Class and methods used to build the application

https://github.com/lana-rodion/p6game_new

26 july 2020

interface.js

Display game rules with toggle button function
function play() and function mute()

index.html

```
<script type="module" src="js/app.js"></script>
<script src="js/interface.js"></script>
```

Notes:
● init() to initialize the game by creating the game grid , to place players, to display accessible cells

● gamePlay() to manage the game turns and to display players description

● playerActions(player, boardCell, cellsAround) to manage the different players actions: to move, to change weapons, to prepare the fight

● prepareClash() to change the appearance of the board before the fight

class Game

import Board from "./board.js";
import { player1, player2 } from "./players.js";
import { weapons } from "./weapons.js";
export default class Game

+ this.turnToPlay = turnToPlay;
+ this.gameBoard = gameBoard;

Methods:
init()
gamePlay()
playerActions(player, boardCell, cellsAround)
prepareClash()
gongSound()
playersDescription(player)

app.js

import Game from "./game.js";

$(document).ready(function() {

$("body").fadeIn(2000);
let game = new Game(true, true);

game.init();

});

Notes:
● createGrid(width, height) defines cell coordinates, to push cells in columns and row with for loop

● randomCell() to return random cell with coordinates x and y, called randomNumber(0, this.width)

● randomPlayers(player) to place random player in random cell, called getAdjacentCells(cell) to verify if adjacent Cells and the cell of player placement are not occupied by other player

● obstacles() inserts the obstacle in random Free Cell

● weaponsArr() to place the weapon in the random Free Cell

● getAdjacentCells(cell) returns all the cases adjacent to a player cell

● getAccessCellsAxis(cell, nbOfAccessCell, horizontal, axis)
returns an array of the accessible cells using the direction

● getAccessibleCells(cell, nbOfAccessCell) concats accessibleCells array to return all cells accessible by the player

class Board

import Cell from "./cell.js"
export default class Board

this.weapons = weapons;
this.player1 = player1;
this.player2 = player2;
this.width = null;
this.height = null;
this.cells = [];

Methods:
createGrid(width, height)
randomNumber(min, max)
randomCell()
players()
randomPlayers(player)
obstacles()
weaponsArr()
randomFreeCell()
getAdjacentCells(cell)
cellExist(x, y)
getAccessCellsAxis(cell, nbOfAccessCell, horizontal, axis)
getAccessibleCells(cell, nbOfAccessCell)

class Player

import { weapon1 } from "./weapons.js";

this.name = name;
this.nickname = nickname;
this.weapon = weapon1;
this.life = 100;
this.currentCell = null;
this.defense = false;

export let player1 = new Player(name, nickname);
export let player2 = new Player(name, nickname);

Methods:
move(newCell)
changeWeapon(player)
isPlayerAround(cellsAround)
heroTarget(target)
heroDefense()
endGameModal()
gongSound()
gameOver()
scoreLife()
fight(target)
restart()

Notes:
● move(newCell) to move player and change the previous cell property

● changeWeapon(player) to exchange the player weapon into the cell weapon

● isPlayerAround(cellsAround) checks if there is a player in cellsAround

● heroTarget(target) to change the appearance of the player who is a target in the fight and to hide buttons

● heroDefense() to give the choice of to attack or defend

● gameOver() to finish the game if one player has not life points and to call modal of endGameModal()

● scoreLife() to calculate life points

● fight(target) to manage the fight, to count fight damages on click

class Cell

export default class Cell

this.x = x;
this.y = y;
this.element = element;
this.obstacle = false;
this.player = null;
this.weapon = null;

Method:
isFree()

class Weapon

export let weapons = [];

this.name = name;
this.damage = damage;
this.nickname = nickname;

export let weapon1 = new Weapon(name, damage, nickname)

Note:
● isFree() checks if this cell is not occupied by an obstacle or a player