Bomberman game by Ilya Mochalov.

1) Description

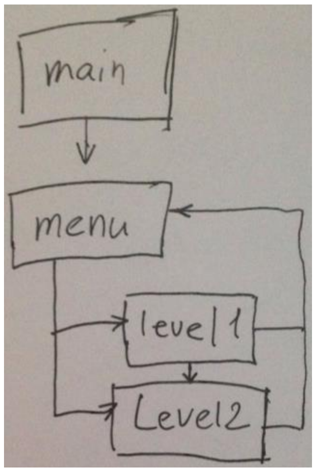
This game has 2 levels, originally second level is locked and becomes available only after player completes first one. To complete level player need to kill all the mobs on level and find entrance to the next level (entrance is hidden under a removable obstacle and looks like a blue bird).

2) Gameplay

We use slide movements to move our player up, down, left, right. To place a bomb, we use double tap gesture.

3) Composer

Basic scene flow is presented below:



4) Modules and scenes

Scenes **menu, level1, level2** use modules **menumeta, level1meta, level2meta** respectively. These three scenes use modules to create game grid system and place the objects (walls, removable obstacles, player, mobs) on the map according metadata. Scenes have functions that implement game logic for particular level (right now game logic on level1 and level2 just the same, so these two files are almost identical and differ only in part where they load levelXmeta module and “finish level” part)

5) Modules description(level1meta)

table M.map holds indexes for our level objects.

table M.nextL holds indexes of entrance to the next level

function M.createGrid determines coordinates for our grid system, later we will use this coordinates to place level objects and move payer with monsters. We save our coordinates in 2D table *M.grid[][]* with keys *.x and *.y. So later when we crate objects and move them we would refer to *M.grid[][] to know where to place object on our screen*

function M.createObjects creates and inserts level objects. We save player to variable *M.player,* walls to table *M.walls[]*, mobs to talbe *M.mobs[]*, removable obstacles to table *M.removable[]*, also we create object *nextLevel* and save it in variable M.nextLevel. In this function we create objects, save it's indexes in object[i].r and object[i].c that stands for row and column. Then we place objects to level using coordinates we saved in *M.grid[][]*. Creating objects we change *M.grid[][].status* property to be "empty", "mob" and etc. so later we can use it in out game logic moving, killing, exploding objects.

6) Scene description and game logic (level1)

In scene:create part I create background and using module level1meta create game grid system and objects. Then I have a timer that calls function *moveMobs()* infinitely.

MOBS FUNCTIONS

function moveMobs(). Loops through table of our level mobs and calls for the auxiliary function *mobCanMove(i)* (check the description below). Once we know directions where the mob can move we randomly choose one and move the mob there.

function mobCanMove(i). Takes index of the mob in mobs table and returns directions where one particular mob is allowed to move. Answer is returned in table.

function killedByMob(r, c) checks weather the mob running into the player and kills him/her.

PLAYER FUNCTIONS

In scene:create we add event listener *sceneGroup:addEventListener("touch", movePlayer)*

function movePlayer(event) utilizes algorithm that tells us weather we can move our player.

function canMovePlayer(direction) returns true or false depending on whether there are obstacles in the way of our player.

BOMB FUNCTIONS

In scene:create we add event listener *sceneGroup:addEventListener("tap", placeABomb)*.

function placeABomb(event) create bomb object and starts timer *bombTimer.*

function bombOnTimer() decreases *bomb.countdown* variable and once it's less than zero we create explosion object in table *explosion[].* While we creating *explosion[]* we also chaek weather explosion kills or destroys anyone/anything (here we call for axillary functions) or even kill the player. Once we created *explosion[]* we set timer *explosionTimer* that would delete the *explosion[]* from our game view.

function explosionOnTimer() removes *explosion[]*

function killMobs(side) using *side* variable passed to the function and for loop we find the mob that is killed by bomb explosion *function explodeRemovable(side)* pretty much like *function killMobs(side)* but about reoving obstacles

ADDITIONAL FUNCTIONS

function nextLevel() check weather all the mobs are killed and player stays at the entrance to the next level, if true then we change our scene

function gameOver( ) goes to game over scene once our player is killed

function handleButtonEvent_menu(event) and *function handleButtonEvent_pause(event)* are self-explanatory

scene:hide we stop all the timers, remove all the objects, unpack modules and remove scene.