Adding functionality guide

In this tutorial we will explain how to add functionality to the game.

Adding functionality consists of three main changes:

- Defining the trigger of the functionality in the client
- Adding a function to the server
- Adjusting the GUI in the client
- Adding rules of validation to the validator.js file

In order to explain it I will demonstrate how to add the functionality of making more than one step in a given direction (up/down/right/left).

Defining the trigger of the functionality in the client:

Moving on board is done by pressing on one of the trails adjacent to our player, now we only need that the trail and the player will be on the same line, this kind of functionality if implemented in the client side (as a principle everything that can be checked in the client will be there).

When the board is built, for each of the trails we defined what will happen upon clicking the trail.

In order to change it we will need to go to the function **beginFaze** (in each phase we check if the board has changed). Inside the function we create the board object and define its properties including the step. We can see there that clicking a trail we check if the player can move, and if he can we check if it's only one step.

So first we need to change the validation:

```
var op1 = (Math.abs(App.Player.locations[App.Player.myid][0]-data.row) === 0) &&
          (Math.abs(App.Player.locations[App.Player.myid][1]-data.col) === 1);

var op2 = (Math.abs(App.Player.locations[App.Player.myid][0]-data.row) === 1) &&
          (Math.abs(App.Player.locations[App.Player.myid][1]-data.col) === 0);

if(op1 || op2)

{make the move}
```

Into this:

```
var op1 = (Math.abs(App.Player.locations[App.Player.myid][0] -data.row) === 0) &&
          (Math.abs(App.Player.locations[App.Player.myid][1] -data.col) >= 1);

var op2 = (Math.abs(App.Player.locations[App.Player.myid][0] -data.row) >= 1) &&
          (Math.abs(App.Player.locations[App.Player.myid][1] -data.col) === 0);

if(op1 | | op2)

{make the move}
```

Adding the function to the server:

After changing the client's code we need to configure the server side, although the step was checked in the client side the server must approve it. There is a call to **MovePlayer** function in the server, this is the function that give the final approval.

The function check again everything the client did to remove the threat of inconsistence so we need to check again. We first check other things too from the server perspective like if the new block is a goal and if there are other game terms as well.

We need to implement the function that take the chips from the player according to the movement. Changing it so he will pay chip for each trail he went through will look like:

```
Var trailsToJumpX = player.location.x – data1.location[0] ;

Var trailsToJumpY = player.location.y – data1.location[1] ;

Var chipsToPay = [];

Var direction;

if(trailsToJumpX > 0 ) direction = "left";

else If(trailsToJumpX < 0 ) direction = "right";

else if(trailsToJumpY > 0 ) direction = "up";

else if(trailsToJumpY < 0 ) direction = "down";

else direction = "undefined";
```

```
If(direction == "left")
            for(var i=1; i<=Math.abs(trailsToJumpY); i++)
                        chipsToPay[ room.GUIboard[player.location.x-i][player.location.y] ] ++;
If(direction == "right")
            for(var i=1; i<=Math.abs(trailsToJumpY); i++)
                        chipsToPay[ room.GUIboard[player.location.x+i][player.location.y -i] ] ++;
If(direction == "up")
            for(var i=1; i<=Math.abs(trailsToJumpY); i++)
                        chipsToPay[ room.GUIboard[player.location.x][player.location.y-i] ] ++;
If(direction == "down")
            for(var i=1; i<=Math.abs(trailsToJumpY); i++)
                        chipsToPay[ room.GUIboard[player.location.x][player.location.y+i] ] ++;
var canMove = 1;
for( var i=0; I < chipsToPay &&  canMove; i++)
            if(chipsToPay[i] > room.playerList[data1.playerId].chips[i])
                        canMove = 0;
if (canMove) {
            for( var i=0; I < chipsToPay; i++)
                        room.playerList[data1.playerId].chips[i] –= chipsToPay[i];
}
else return;
/**********             end of code             ***********/
```

I checked how many chips should be paid and if the player has the amount, if he has he will move and pay.

Adjusting the GUI in the client

As we can see, next there is the definition of the data sent to each of the players in the room (line 777 in the server), including the new location of the player, so the client will place the player few steps from where he was.

But there is only one chip we mention to pay so it should be changed to an array in the size of the number of colors, in each cell the amount of chips to be paid of the respective color.
We defined the array as colorsToPay[] when we checked the amount of chips to be paid.

Back to the client code:

Function movePlayer() in line 455 decrease one chip from the player according to the chip sent in the data, now it is an array, so we will change it so for each color we decrease the amount mentioned in the respective cell in the array.

Adding rules of validation to the validator.js file

This is an optional procedure, if we would want to make rules for the possibility of making more than one step in a given direction (up/down/right/left), we will need to add the rules to the conf validator (Validator.js).

One of the validator's responsible is to check if the configuration file is legal. So if we want to add field to the configuration file we need to ensure that the validator will accept it.

In other words, if we want to add the possibility to move more than one step in a given direction, we need to decide which field we want to add to the configuration file, and add rule of validation in the Validator.js file.