1. How the game is played.

Let’s talk about the gameplay() function. This function is called basically 25 times in one second and it is to calculate the new position of the player. You can find out that at the end of this function, updateScreen() is called, so we can infer that updateScreen() is called 25 times in one second. This means that we do not need to call updateScreen() on purpose if we want to change the display of the screen, because it is regularly called 25 times a second, which is a very fast frequency. But if you want to call updateScreen() on purpose, it is OK.

1. Generally speaking, we do two things in transformation.

The first part is scale(sx, sy), in which sx and sy are scaling levels along x axis and y axis. In other words, they are assigned by the zoom variable. The zoom variable is initially set to 1 when no zoom in is applied.

The second part is translate(tx, ty). Why do we need to do the translation? Let’s look at the picture below.

W

B’

B

x

D’

C’

C

A

H

F

E

D

G

H

y

Assume the scaling level is two. The original screen is ABCD and after scaling the screen becomes AB’C’D’. The player also changes its position. Remember our objective is to put the player in the middle of the screen (which size should be the same as ABCD). Therefore we choose an area EFGH to locate the player in the middle. Then we should move the rectangle EFGH to the original place ABCD, because only stuffs in ABCD can be displayed and seen. How far should we translate, i.e. what are the values of tx and ty?

We assume that the coordinate of left-top corner of the player is player.position.x and player.position.y. After scaling, the center of the player should be ((player.position.x+player.WIDTH/2)\*zoom, (player.position.y+player.HEIGHT/2)\*zoom). So we can infer that tx = -(W/2-(player.position.x+player.WIDTH/2)\*zoom) and

ty = -(H/2-(player.position.y+player.HEIGHT/2)\*zoom).

So we use translate(tx, ty).

But there are two special cases.

1. Let’s consider what if the (player.position.x+player.WIDTH/2)\*zoom<W/2 or (player.position.y+player.HEIGHT/2)\*zoom < H/2. In these conditions, we cannot put the player in the middle. Then we just leave tx = 0 and/or ty = 0. See graph below:

W

x

D’

C’

C

A

H

D

1. What if (zoom\* W - (player.position.x+player.WIDTH/2)\*zoom)< W/2 or

(zoom\*H - (player.position.y+player.HEIGHT/2)\*zoom)< H/2? We set tx = -(zoom\*W-W) and/or ty = -(zoom\*H-H). See the graph below.

W

x

D’

C’

C

A

H

tx

ty

1. What to do with the zoom button?

<g onclick=“top.setZoom()" transform="you are not required to specify this!">

<rect x="0" y="0" width="140" height="40" style="fill:white;stroke:red;stroke-width:2" />

<text x="70" y="28" style="fill:red;font-size:25px;text-anchor:middle">Zoom</text>

</g>

First I want to remind you that when calling a function in the .js file from the .svg file, we should add “top.” before the function.

Then, what to do with the setZoom()? Basically, you just need to assign zoom a zoom level, say 2 or 3 or something. And you also need to use this zoom level in the updateScreen function, just like what I did in the second part! Actually after this, whether to call the updateScreen function is up to you. For the reason, you can refer to part 1.