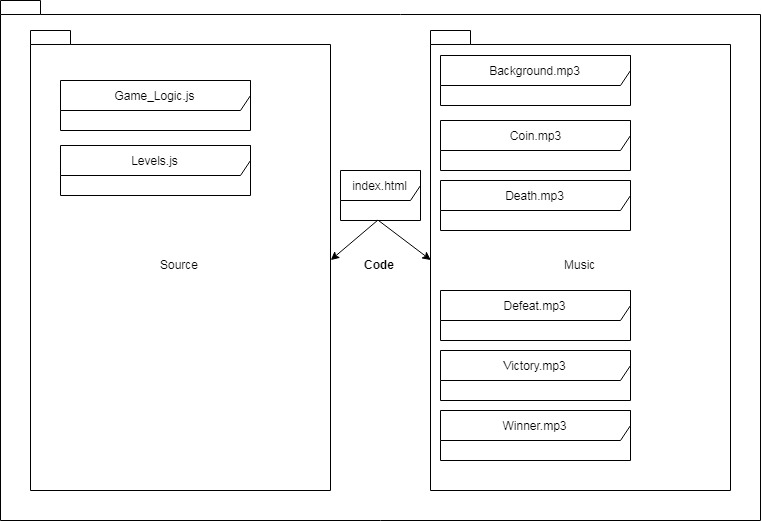
Iteration 3

UML Diagram: Package Diagram of Current Code



Iteration Work Plan:

* The Goal of The Iteration
  + Uses coloured blocks as images
  + Plays in less than 180 seconds
  + Not be offensive!
  + Sound
* The Planned Tasks in Sequence [Planning, Analysis, Design,

Coding, Testing]

* + Uses coloured blocks as images - E
  + Plays in less than 180 seconds - F
  + Not be offensive! - G
  + Sound - H
* A Time Estimate for Each Task [30 Minute Blocks]
  + E – 30 Minutes
  + F – 30 Minutes
  + G – 30 Minutes
  + H – 30 Minutes
* The Planned ‘Product’ Of Each Task
  + E – Uses Coloured Images as Blocks
  + F – Plays in less than 180 Seconds
  + G – It’s not Offensive
  + H – Sound Working
* A Record of The Actual Time Each Task Took
  + E = 0:05:00
  + F = 3:19:00
  + G = 0:01:00
  + H = 2:50:00

**PLANNING A COMPLEX ALGORITHM**

**DESIGN THE ROUTINE**

CHECK PREREQUISITES

Define the problem

*Counting Down from X time when the game starts*

Outputs from the routine

How much longer the player can play before they time out

Pre-conditions

*The game is running*

Post-conditions

If time left === 0, then reset the level

Decide how to test the routine

The time should display properly on the screen

Research functionality available in standard libraries

*Can do it natively using the promise function type*

Think about error handling

Math.floor will return 0 if something goes wrong, so we should be fine

Think about efficiency

Needs to be efficient since it is called on each frame

Research algorithms & data types

*Need to use Math.floor to get whole integers, need to use Promises’ time feature to get current time*

**WRITE PSEUDOCODE**

For each frame the game processes (in the level), add the time difference between each frame to a variable,

Then to get remaining time we Math.floor the total allowed level time (can change with level) minus how long has passed in the level

**CODE THE ROUTINE**

function runLevel (level, Display) {

let display = new Display(document.body, level)

let state = State.start(level)

let ending = 1

// FOR COUNTDOWN TIMER

let elapsedTime = 0

let remainingTime = 500

let playedDeathSound = false

//

return new Promise(resolve => {

runAnimation(time => {

state = state.update(time, arrowKeys)

// FOR COUNTDOWN TIMER

elapsedTime += time

remainingTime = Math.floor(state.level.completeTime - elapsedTime)

if (remainingTime === 0) {

state.status = 'lost'

}

if (state.status === 'lost' && !playedDeathSound) {

playSoundWithID('Death')

playedDeathSound = true

}

display.setState(state, remainingTime)

//

if (state.status === 'playing') {

return true

} else if (ending > 0) {

ending -= time

return true

} else {

display.clear()

resolve(state.status)

return false

}

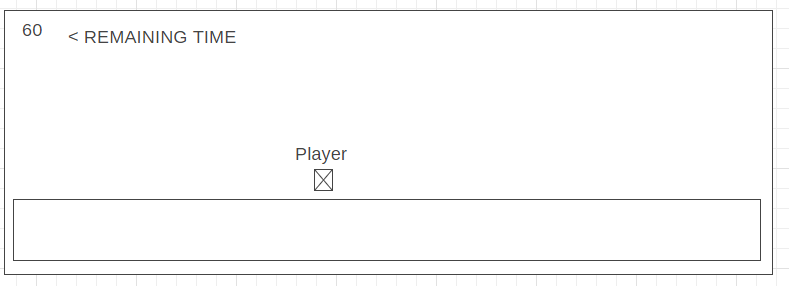
**CHECK THE CODE**

Working Perfectly!

A plan for how the program feature you are working on will work [UML dynamic diagram, story-boards, wireframe, pseudocode]:

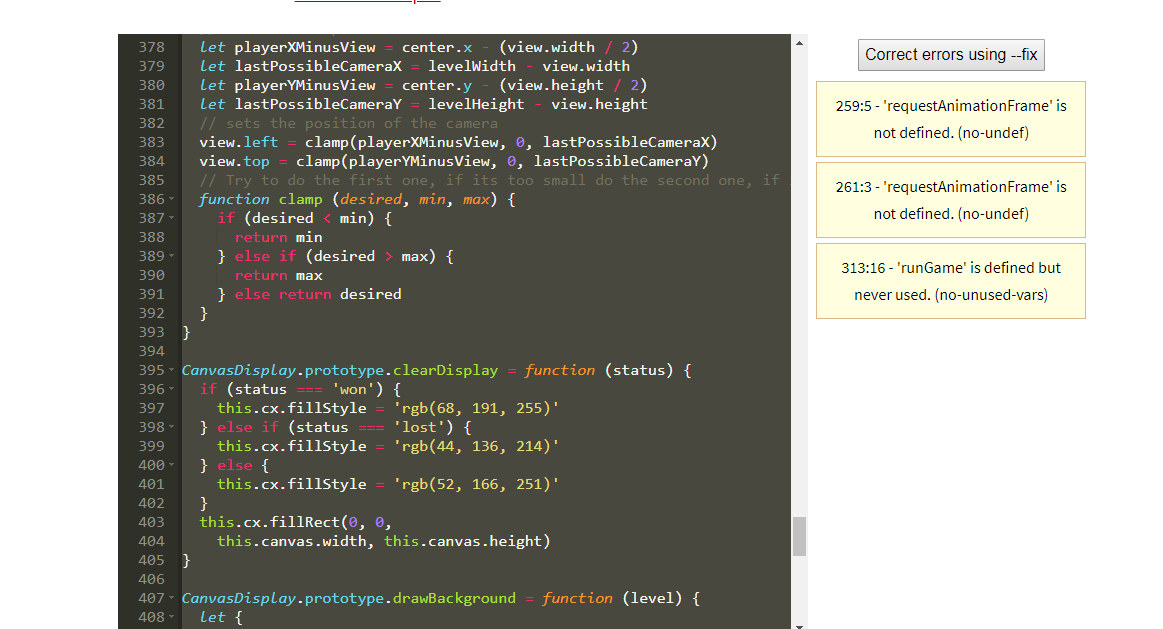
Sound will be played when the user is playing the game (background), when they collect coins (coin), when they die (death), when they clear a level (Victory), when they win the game (Winner) and once the lives mechanic is implemented, when they lose the game (Defeat).

The player has 60 seconds to complete a level, once those 60 seconds run out the level is restarted, but further on I want the player to lose lives at that point.



POST CODE COMPLETION>>>>>

A report showing nil style defects in your code according to JavaScript Standard Style https://standardjs.com/index.html:



Mistakes were made! A description and analysis of the mistakes made in the iteration:

The timer took a lot longer than I expected due to a few efficiency problems, so I had to split what was originally Iteration 2 into Iteration 2 & 3

The sound was initially easy to setup, but I found sourcing good sound files a pain (I used audacity to create the Coin sound file and sourced the rest of them from YouTube)

Also, for sound to combat bad websites spamming their audio files chrome has disabled the autoplay feature (video and audio) on their browsers, so I had to work around that with a play button

Lessons were learned? A plan for doing ONE thing differently in the next iteration:

Make sure to comment all the code and separate classes into their own .js files