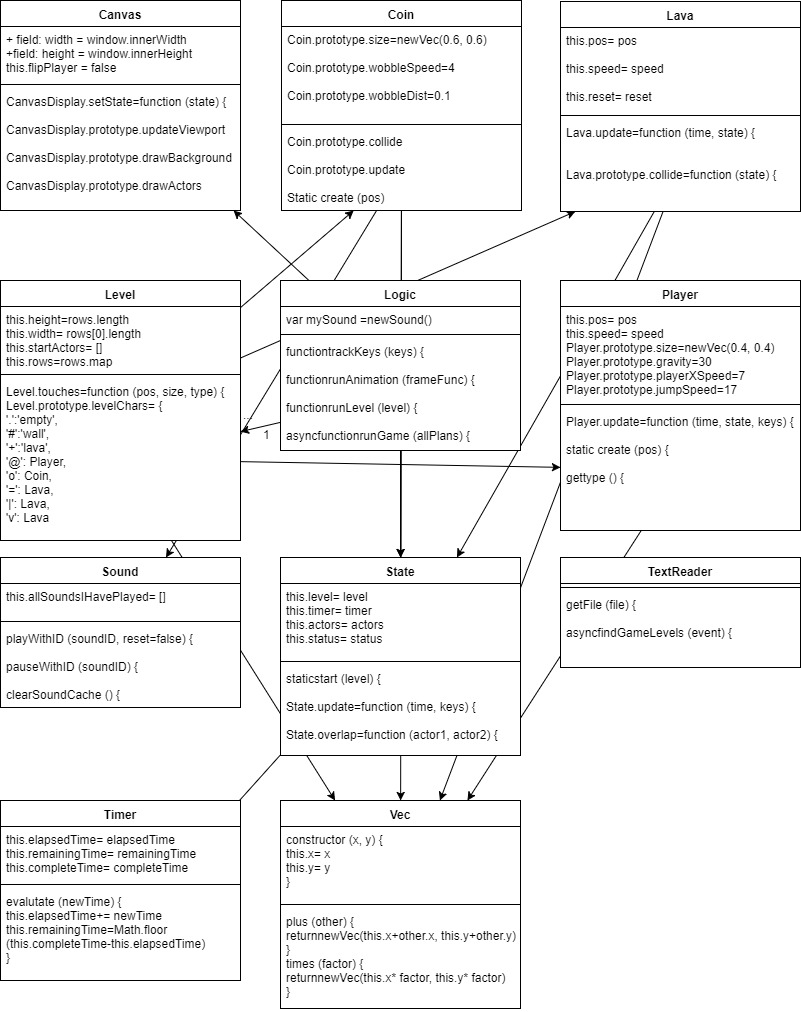
Iteration: 5

UML Diagram 1: Class Diagram of Starting Product (As of Iteration 5)



UML Diagram 2: Class Diagram of Finished Game

Iteration Work Plan:

* The Goal of The Iteration
  + Can Play on Mobile
  + Lives
  + A theme for the game relevant to the Christchurch rebuild. (a saying, a person, an identifiable place)
  + Uses images derived from the Christchurch Rebuild.
* The Planned Tasks in Sequence [Planning, Analysis, Design,

Coding, Testing]

* + Can play the game using a mobile device – A
  + Lives – B
  + Thematic - C
* A Time Estimate for Each Task [30 Minute Blocks]
  + A – 120 Minutes
  + B – 60 Minutes
  + C – 120 Minutes
* The Planned ‘Product’ Of Each Task
  + A – Playable game on a phone
  + B – Lives work
  + C – Fits with the Christchurch rebuild theme
* A Record of The Actual Time Each Task Took
  + A –
  + B – 3:01:30
  + C –

**PLANNING A COMPLEX ALGORITHM**

**DESIGN THE ROUTINE**

CHECK PREREQUISITES

Define the problem

The player should start the game with 3 lives and lose a life each time they die.

Information the routine will hide

Will contain and evaluate player lives

Inputs to the routine

How many lives does the player start with

Outputs from the routine

How many lives does the player have now

Pre-conditions

The game is running, the lives do not reset over the course of the game

Post-conditions

There are still lives

Name the Routine

reducePlayerLives()

Decide how to test the routine

The Lives will display on the screen

**WRITE PSEUDOCODE**

When the player dies they should lose a life, if they are out of lives, they lose the game

The sounds, timer and player position should reset when the player dies, but not the coins

If we lose the game, go back to the main menu and play a special sound clip

**CODE THE ROUTINE**

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

state.status = state.player.lives.reduceAndCheck()

if (state.status === 'playing') state.player.reset(state)

mySound.clearSoundCache()

state.timer.reset()

}

/\* global mySound \*/

var Lives = class Lives { // eslint-disable-line no-unused-vars

// ^^^ Just becuase we're hosting this in a different file so its actually used globally

constructor (startingLives) {

this.count = startingLives

}

reduce () {

this.count--

}

reduceAndCheck () {

if (this.count === 0) {

mySound.playWithID('Defeat')

return 'lost'

} else {

this.count--

mySound.playWithID('Death')

return 'playing'

}

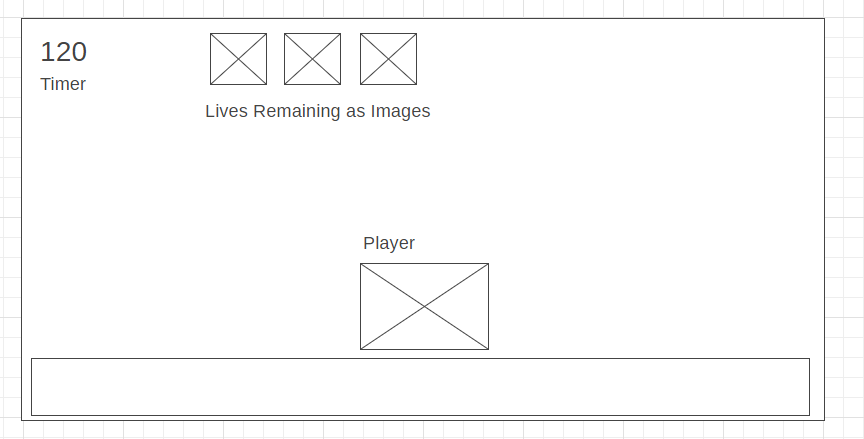
}

}

**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]:



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:

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