Questions:

* How does overwriting changes/commitments work?

Game functions:

**void waitToStartGame() 79**

// wait and do nothing until some hovers one hand above sensor

//then continue with game loop

**void checkIfGameOver() 97**

// check if user won or lost

* Is this function complete?

**void updateTarget() 104**

//choose a random number from 0 to 3

// increase oldTarget (pin) by the number and setting it as the new targetPin

// if(targetPin > NUMTARGETHOLES) ?? (line 114)

// close old target light and opens new target light (updateLights function)

**void resetBar() 121**

//given current position of bar, reset the bar

//put the ball back onto the bar

**void resetBall() 127**

//wait until ball is ready to roll onto the bar

//put ball back onto bar

**void resetGame() 134**

//resetBar, resetBall

//initialize score, finishTime

**int smooth\_distance(int num\_samples) 146**

**int sample\_distance() 167**

**void moveBar() 196**

// get user input (ie move left down by x amount)

// control motors to move bar

// check position of bar (don’t move if bar is tilted, at top, or at bottom)

**bool beamBroken(int target) 207**

//check if sensor beak is broken (if yes, sensorState == LOW)

**void ballEntry() 249**

// if targetBroken is true, ball fell in good hole, updateScore

// if bottomBroken is true, ball fell in wrong hole, game is reset (resetGame)

**void updateLights(int lastHole, int newHole) 276**

// pretty intuitive…

**void updateScore() 296**

// uses time user takes and difficulty of hole to update score

* need to update score on seven seg display too?

**void displayMaxScore() 312**

// ah, here

//3 digits?

**void setup() 317**

//setting up here… not too sure how this works

**void loop() 345**

// hello game loop

* score and time are both updated in resetgame
* score updated in ballentry

powerdowns

* from level 0 to 4 🡪 on power down, intro to game
* from 5 onwards 🡪 in increments of 5 (5, 10, 15, …), there will be an increase in the combination of randomly selected powerdowns