[ons feb 13 14:18] Initialized GitHub and joined our git project (30 min of work)

[mon feb 18 10:00-17:00] The three of us, together, decided to use 'gloss' as our way of displaying graphics. For about an hour I reserched about what possible ways there are to display grids or rows and columns. We then decided that the easiest way we can get started with using gloss would be to have every individual square be represented as a rectangle picture within gloss. We created a grid of 5*5 squares. At about 13:00 after lunch I started researching how to create a gameloop. For about an hour I tried using the built-in gloss playloop, without success. After that the three of us started trying individual strategies in hope that one of us would stumble upon a fitting way to create a gameloop. I tried using the 'do' notation, but none of the ways i tried to create a game-loop felt fitting. We spent a really long time trying to figure out how to create a fitting game-loop until one of us succeded with the built-in gloss playloop within a 'do' notation. Thats where we ended the day, at about 17:00-18:00. At this point we had a playing field. FUNCTIONS MADE main render initState squareLocations

[tue feb 19] My ssd gave up. I spent the whole day trying to reformat and debug the ssd, only to find out that every single block on the entire ssd had failed. The ssd gave up at 10:00 and I gave up, admitting defeat, at 18:00. A new ssd is on the way, right now I´m borrowing a friends laptop.

[ons feb 20 12:00-20:00] Alot happened when I was unable to work, we now have a 'player' (a square that can be moved using the arrow-keys) and we have 'candy' on every square (another rectangle within every rectangle). Today I managed to make the playing field dynamic, in other words, the amount of squares on screen can be changed to whatever square of a natural number. After solving this, I decided to tackle random numbers, we're going to need random numbers in order to generate random 'candies'. After a couple of hours of trying to convert an IO Int into a regular Int I decided to use the 'unsafePerformIO' preliminarily. We can now randomize the color for ALL candies, but not for individual candies. This is probably because haskell is a lazy language and doesn't want to re-evaluate something it doesn't have to. I found a possible solution: ´deepseq´. I didn't have to to implement it since it had gotten late. FUNCTIONS MADE createCandy paintCandy recolor

[thu feb 21 12:00-21:00] We realized that our datatypes didn’t support the way we wanted to handle candies. Therefore I created a new datatype Candy = (((Float,Float),Int),Color) that contained everything we wanted every separate candy to contain. After I implemented this new datatype I could rewrite our ’move’ function (handles the way the player moves across the playing field) to completely depend on which candy the player is currently sat at. Before this change the player moved by adding and subtracting from the players current coordinates. When the new datatype was implemented and up and running I created a function for swapping candies. At about 21:00 the function was bug free and working.

FUNCTIONS MADE
datatype Candy = (((Float,Float),Int),Color)
moveSquare (remade)
moveCandy

moveCandyAux

[fri feb 22] Day off

[sat feb 23 5h] Created a function that checks whether or not there is three candies or more in a row (horizontally). If there is, these candies are colored black. Created seperate gameStates, can be changed by pressing 1 and 2 (preliminarily)
FUNCTIONS MADE
checkRows
handleKeys (remade)
render (remade)

[mon feb 25 7h] Completely remade checkRows, and added two auxillary functions that check candy-rows vertically and horizontally. Created functions makeBlackV and makeBlackH, these functions are used for coloring candies in a row black, both vertically and horizontally. After being colored another function moves all black candies to the top of the gameBoard where they then will be recolored into a new random color. (not yet implemented)
FUNCTIONS MADE checkRows (remade)
checkHorizontalRows
checkVerticalRows
makeBlackV
makeBlackH

[tue feb 26 7h] Created functions to recolor all black candies into new random colors.
FUNCTIONS MADE
mkAllCol
makeAllColor
recolor
getColor

[wed-thu feb 27-28 16:00 - 06:00 (all night)] Wrote test-cases for functions: updateScore squareLocations paintRectangles getColor randColorGen randListGen mkMarker candyLocations paintCandy createCandy moveBlackAux Wrote following subsections in our projekt report: render verifySwapCandy and verifyMoveCandy moveCandy Made a picture for every important function and algorithm we wrote about in the project report (all pictures)

[fri feb 29 4h] Wrote following subsections in our project report Datatype Use cases

[mon mars 4 2h] Wrote following subsections in our project report Conclusion