Byron Huang for David’s Game

1. Describe each game feature from a technical perspective
   1. Identify programming difficulties of each feature
      1. Constantly updating the running motion of the character
      2. Randomly set up obstacles on the screen and the interaction of the character with said obstacle
      3. Randomly set up power ups on the screen and the interaction of the character with said power up
      4. With the obstacle banana peel, the feature of losing control acts upon the character
      5. With the power up diamond, having immunity from obstacles for 10 seconds
      6. The transition from one level to another
   2. Define / pseudo-code core functions each feature will need
      1. Character
         1. moveRight()
         2. moveLeft()
         3. jump()
         4. slide()
         5. lives()
      2. Game Feature
         1. coins()
         2. bananaPeel()
         3. heart()
         4. diamond()
         5. hurdle()
         6. bridge()
         7. background()
         8. immunity()
         9. loseControl()
         10. gainLife()
      3. Levels
         1. nextLevel()
      4. Performance
         1. timer()
         2. tokenCount()
         3. lifeCount()
   3. Include the performance impact of each feature
      1. All the features will be relatively quick, there is nothing that will cause it to lag or slow the game down.
2. Identify dependencies or assumptions
   1. Does this code need any libraries or rely on pre-written code?
      1. No, this game is not too complicated so there is no need for libraries or pre-written code
   2. Are there design assumptions which are actually entire features?
      1. No extra coding is required
3. Prioritize each feature (with reasoning)
   1. Getting the background moving and the random obstacles spawning. This will probably be the hardest part because this is the layout of the whole game.
   2. Interaction of the character with the obstacles/powerup. This will be hard because each obstacle/powerup has a different effect on the character.
   3. Losing lives, gaining coins etc. This also shouldn’t be too hard because it’s relatively easy to code up. If the character touches a coin, it gains a coin.
   4. Leveling Up. This shouldn’t be that hard because the screen will only move faster.
4. Feasibility analysis
   1. What are the most difficult items to program regarding technical difficulty?
      1. It will be difficult to program the interaction of the character and the randomness of the obstacles spawning. When you jump, the character has to look like it’s jumping. You also need the obstacles to spawn at random times but it has to have good timing otherwise sometimes there can be a bunch of obstacles spawning at once.
   2. What are the commitments regarding time needed to program?
      1. It will probably take a long time to design each feature of the character and the interaction of the obstacles and powerups with the character.
   3. Identify technical issues with the design
      1. If a character trips on a banana peel, the player loses control for 5 seconds, that’s way too long for a game with many obstacles. The diamond power up also gives way too much immunity time. There is only 60 seconds per level and 10 seconds of immunity is too good.
5. Data storage and variables
   1. Layout and list all variables
      1. Character’s lives
      2. Time Alive
      3. Number of Coins collected
      4. Position of character
   2. Label functionality of each variable
      1. The character’s lives determine whether or not the player loses the game
      2. The time alive is used for either in multiplayer, to see which player survives the longest, and also for whether you lose the game, because you lose the game if you don’t collect 5 coins within 60 seconds.
      3. Number of coins collected is what the name suggests. If you collect 5 coins then you can move onto the next level. It is also used in multiplayer because the more coins you collect, you win.
      4. Position of character determines which lane the character is on the screen
   3. Call out any data structures needed for program operation
      1. There needs to be a hash which has the obstacles/power ups as the keys and the character as the value to determine whether or not the character is interacting with the obstacles/power ups.
   4. Identify which data is global and which is local
      1. All of these data are global variables
6. Functional code relational map
   1. How do functions relate to one another
      1. All the obstacles and power ups like the bridge or the banana peel have to be randomized and checked whether or not it is hitting the character during the game. It also has to make sure that there aren’t more than one obstacle overlapping each other.
   2. Which functions are called for which part of the program
      1. During the game, all the functions that are random like generating a banana peel or a bridge are all constantly running during the program. Functions that involve interaction like running through a coin will only run when the character is interacting with the coin in game.

Design Issues:

Making the immunity time due to the power up of diamond last 4 seconds instead of 10

Making the “lose control” feature from slipping from the banana peel 3 seconds instead of 5.

Programming Schedule:

Day 1-4: Make the **vertical and horizontal movement** along the screen. Vertical movement is at a constant speed and horizontal movement is controlled by the user. If extra time, then create the **sprites/images** of the characters, obstacles, tokens and power ups.

Day 5-8: Finish the **sprites/images** of the characters, obstacles, and power ups. Create the **background** and the **random generation** of the obstacles and power ups on the background

Day 9-13: Create the **interaction features** of the characters with objects such as the obstacles and power ups. Ex: If they touch one of the power ups like the diamond, they are immune for 4 seconds so hitting any other obstacles will have no effect. Do this for all obstacles and their corresponding outcomes.

Day 14-16: Finish up every other component like the **different levels, lives, and token count.**