Criterion E: Evaluation

Success Criteria

- A – well met. The client thought the controls were intuitive.
- B – well met. The client believed there was a focused camera on the **Character** while moving, though actually it is an illusion since it is the Map that moves.
- C – works. While playing, the client noticed that his **Character** could not move out of the Map.
- D – works well. The client noticed that dots were spawning randomly with different colors.
- E – well met. When the **Character** is in contact with a dot, the dot will disappear, and the character will gain in size. The client also noticed that his **Character**’s speed also decreases, following the mechanics of the real game
- F – works. The client believed that the **AI Characters** were well made, and acted similar to real players, except that the **AI** do not seem to attack each other.
- G – works. The client noted that **AI Characters** are moving even when they are not in the field view of the player, meaning that the randomizing algorithm provided a simulation of the eating process for **AI Characters**.
- H – works. The client said the eating certainly works, but in the actual game the eating was smoother, the eaten character doesn’t disappear so quickly.
- I – well met. The client liked the score system that keeps track of the number of dots/characters each character eats and ranks the characters accordingly. He also liked the idea that the game ends when any character reaches 100 points, for an offline version.

Recommendations for Further Development

The boundary sensing algorithm involves the usage of colors to identify the position of the **Character**. While it works, it is not an efficient code design because graphic properties are usually non-universal and differs from platform to platform. A better design is to have four additional wall objects that senses the contact with **Character**.

The AI of the game could be improved by implementing path-finding algorithms, to determine the optimal direction to move in, such as finding the closest dot. Due to Scratch's limited programming capacity, it was difficult to implement an AI that actually interacts offscreen. By exporting the current program to a more high-level language like Unity or Unreal Game engines, more powerful and smart AI could be implemented.

If the dots are not eaten fast enough and when characters all grow too big, the game may lag at some points (due to memory/CPU processing). The lag can be reduced by implementing a load function that waits for all object properties to initialize, before launching the game, in other words, the randomized spawn locations and chances are all determined beforehand.

Word Count 423