Explore – Impact of Computing Innovations
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Computational Artifact

2a)

My program is made in Javascript on Code.org. The purpose of my game is to create a fun, somewhat challenging game for someone to play in their free time. In my video, the player sees the instructions on how to play the game and then proceeds to control the elk who has to collect the cupcakes. For the first attempt, the player fails by losing all of their elk's lives and has to play again by resetting the game. On the second try, the player stays away from the enemies and collects the cupcakes. The video also shows the elk dodging the oncoming barrier trying to push it away. Also, the elk collects hearts to give it more lives.

2b)

I decided to use Javascript on Code.org because it was an easy way to make an interesting game that was made entirely independently. My game was simple but then became much more. First, I set up my elk which could be controlled from the arrow keys. Then I added cupcakes for something the elk needed to get which increased its score. Next, I added three enemies as challenges to decrease the elk's lives. Last, because I wanted to use power-ups I made a heart for extra lives. A difficulty I encountered was that my instructions would show up on my actual game. I resolved this issue through various tests by making another variable called proceed which was only true if the player pressed enter. This would remove the instructions and would call all the other functions for the game to continue. An opportunity I discovered was making a different type of obstacle that wouldn't harm the elk, just try to push it off the edge of the screen. I revised and implemented this idea by deciding to make a barrier sprite that had its x value decrease by five constantly and also using Code.org's displace function.

Computing Innovation

2c)

```
function touchLocation() {
  spriteCollision();
  locationCheck();
}
function spriteCollision() {
  if (trex.isTouching(elk)) {
    elk.x = randomNumber(10,390);
    elk.y = randomNumber(10,390);
    lives = lives - 1;
  }
  if (ptero.isTouching(elk)) {
    elk.x = randomNumber(10,390);
    elk.y = randomNumber(10,390);
    lives = lives - 1;
  }
  if (axe.isTouching(elk)){
    elk.x = randomNumber(10,390);
    elk.y = randomNumber(10,390);
    lives = lives - 1;
  }
  if (barrier.isTouching(elk)) {
    barrier.displace(elk);
  }
  if (barrier.isTouching(cupcake)){
    barrier.displace(cupcake);
  }
  if (elk.isTouching(cupcake)) {
    cupcake.x = randomNumber(10,390);
    cupcake.y = randomNumber(10,390);
    score = score + 1;
  }
}
function locationCheck() {
  if (elk.x<-10) {
    lives = lives - 1;
    elk.x = randomNumber(10,390);
    elk.y = randomNumber(10,390);
  }
  if (cupcake.x<-10) {
    cupcake.x = randomNumber(10,390);
    cupcake.y = randomNumber(10,390);
  }
}
```

The algorithm I choose is my touchLocation function. In this algorithm, there are two sub-algorithms that are called. The first being spriteCollision contains different if-then statements that check if one character is touching another. For example, if the three enemies are touching the elk in the first three if statements, the elk will reset to a random location and it will lose a life. The barrier that goes from left to right in the fourth if-statement displaces the elk while in the fifth-statement the barrier displaces the cupcake. Also if the elk touches the cupcake in the sixth if-statement, the score increases by adding one to the score variable, and the cupcake's location changes. The second sub-algorithm is locationCheck which checks if the sprites go off the screen. In the first and second if statements the elk and cupcake's location is checked if the barrier pushes it out of the screen and if that happens then the elk and the cupcakes location will reset to a randomly generated location on the screen. Together these algorithms form the main control center of the game which controls all collisions and the effect of those collisions, hence the main algorithm name touchLocation.

2d)

```
function heartRandom(){
  var x = randomNumber(1,500);
  showHeart(x);
}
function showHeart(x){
  if (x == 5){
    heart.visible = true;
  }
  if (elk.isTouching(heart)){
    lives = lives + 1;
    heart.x = randomNumber(0,400);
    heart.y = randomNumber(0,400);
    heart.visible = false;
  }
}
```

The abstraction that I made independently was the showHeart function which is constantly called in the heartRandom function. The heartRandom function constantly makes a random number from one to five hundred and that number is assigned to a variable called x. This variable is then sent as a parameter to the showHeart function where if it equals five then the heart will appear. Also in this function, if the elk touches the heart then the heart disappears, and the random number counter starts over. Touching the heart gives the elk one more life by adding one to the variable heart. This abstraction helped to manage the complexity of the program because if showHeart wasn't constantly being called in the heartRandom function, then I would have to type each individual line of code for the heart to appear and be taken by the elk. The way it is right now, it is very organized and the random number is being made in one function and then sent to another function where it is being used. Fewer lines of code are a result of the abstraction and the randomness of the heart appearing gives the game a different aspect.

References

2e)