Create – Applications From Ideas
Written Response Submission Template

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

Program Purpose and Development

2a)

The programming language I used is javascript on Game lab (Code.org). Code.org uses "functions" which are the equivalent to "methods." The purpose of my program is to be a game that provides entertainment to the user. My video demonstrates the plane moving right and left to catch the animals and the score increases with each animal the plane catches. If one of the animals passes the plane and touches the clouds below the plane, the game ends, displaying the player's score.

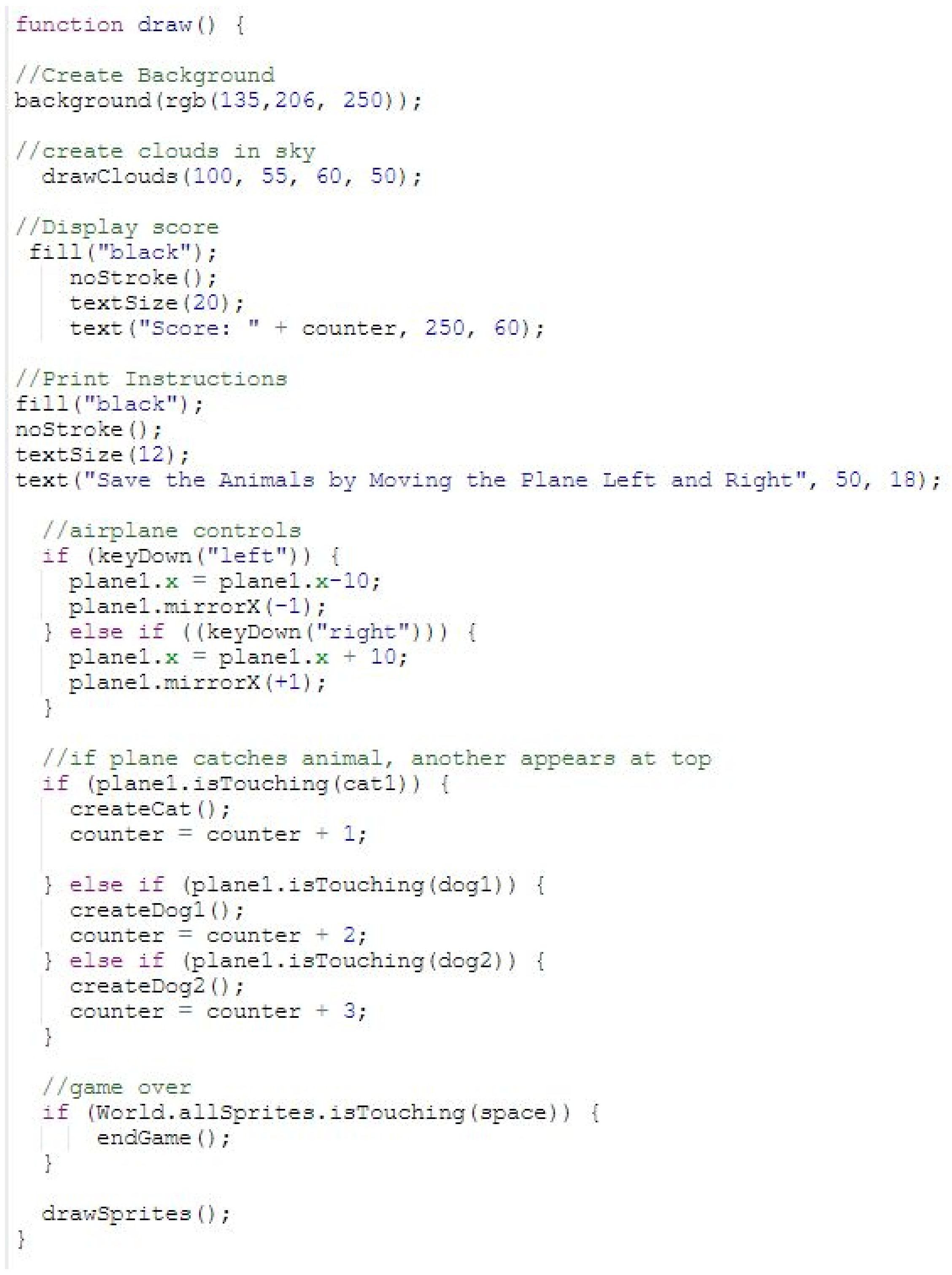
2b)

The incremental and iterative development process of my program included creating the background and clouds first, designing the sprites and adding them to my code, then coding their movement. After testing that, I had to code the airplane to move left and right on the screen when arrow keys are pressed and code the animal sprites to appear as if they are "falling."

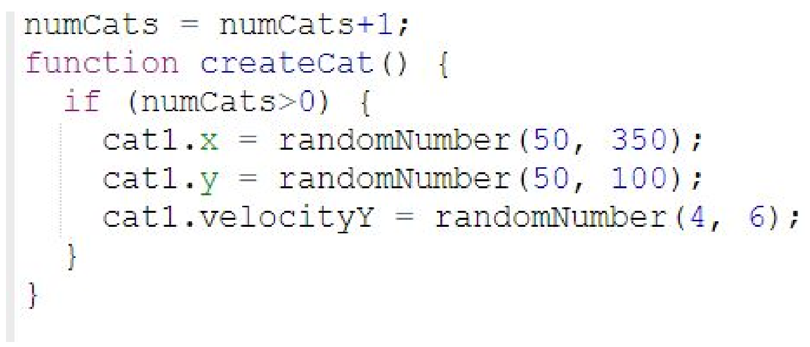
One of the difficulties I had when moving the plane was that the animation of the plane made a trail of several planes so my code ended up printing every single frame. After reflecting on why this occured, I independently solved this problem by using a "draw()" function. In Game lab, the "draw()" function allows sprites to iterate through each frame smoothly, displaying a natural movement look instead of printing out each frame individually. I then asked a classmate to test my game and give feedback.

When getting feedback from my peers, the opportunity of changing the "fall speed" of animals and adding a scoreboard arose, so I changed the velocity to a random number while collaborating with my classmate and added a scoreboard variable and counter independently. Lastly, I added a "game over" screen independently.

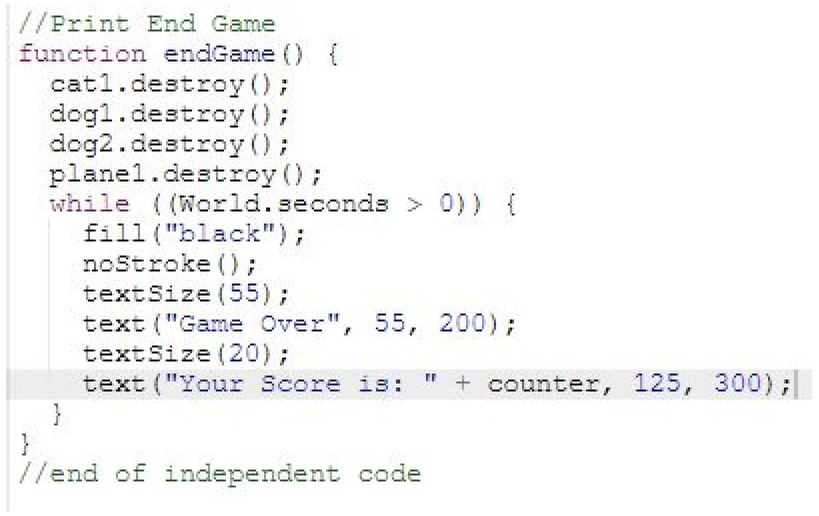
2c)



```
function draw() {

//Create Background
background(rgb(135,206, 250));

//create clouds in sky
  drawClouds(100, 55, 60, 50);

//Display score
 fill("black");
    noStroke();
    textSize(20);
    text("Score: " + counter, 250, 60);

//Print Instructions
fill("black");
noStroke();
textSize(12);
text("Save the Animals by Moving the Plane Left and Right", 50, 18);

  //airplane controls
  if (keyDown("left")) {
    plane1.x = plane1.x-10;
    plane1.mirrorX(-1);
  } else if ((keyDown("right"))) {
    plane1.x = plane1.x + 10;
    plane1.mirrorX(+1);
  }

  //if plane catches animal, another appears at top
  if (plane1.isTouching(cat1)) {
    createCat();
    counter = counter + 1;

  } else if (plane1.isTouching(dog1)) {
    createDog1();
    counter = counter + 2;
  } else if (plane1.isTouching(dog2)) {
    createDog2();
    counter = counter + 3;
  }

  //game over
  if (World.allSprites.isTouching(space)) {
      endGame();
  }

  drawSprites();
}
```



```
numCats = numCats+1;
function createCat() {
  if (numCats>0) {
    cat1.x = randomNumber(50, 350);
    cat1.y = randomNumber(50, 100);
    cat1.velocityY = randomNumber(4, 6);
  }
}
```



```
//Print End Game
function endGame() {
  cat1.destroy();
  dog1.destroy();
  dog2.destroy();
  plane1.destroy();
  while ((World.seconds > 0)) {
    fill("black");
    noStroke();
    textSize(55);
    text("Game Over", 55, 200);
    textSize(20);
    text("Your Score is: " + counter, 125, 300);
  }
}
//end of independent code
```

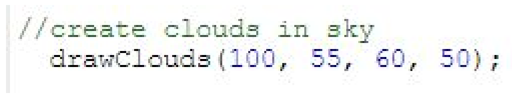
The main algorithm I chose is the “draw()” function and it makes a call to two other methods: “createCat()” and “endGame()” to make the sprite animations work in the game.

“createCat()” is programmed to randomly select the number for the x and y-coordinates and the velocity of the cat object by checking if the number of cats is greater than zero and if it is true, the sprite’s x and y coordinate are chosen by a randomNumber between 50 and 350, and 50 and 100 respectively. Then the cat’s velocity is chosen between a randomNumber between 4 and 6.

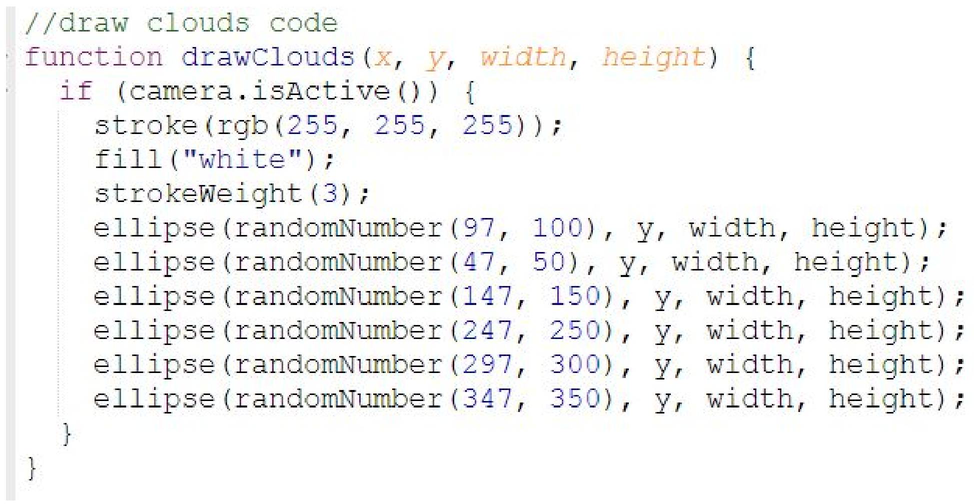
The second algorithm is “endGame()” which is called when an animal touches the clouds below the airplane. This function works by checking if the sprite touches the clouds then removing all the sprites from the screen. While the seconds elapsed from when the game started is greater than zero, the “Game Over” text and score will be displayed.

Together, these algorithms help create a game that provides entertainment to the user by making the animation seamless, adding cats to allow the user to earn points, and displaying a “Game Over screen” when the user fails to save one of the animals.

2d)



```
//create clouds in sky
  drawClouds(100, 55, 60, 50);
```



```
//draw clouds code
function drawClouds(x, y, width, height) {
  if (camera.isActive()) {
    stroke(rgb(255, 255, 255));
    fill("white");
    strokeWeight(3);
    ellipse(randomNumber(97, 100), y, width, height);
    ellipse(randomNumber(47, 50), y, width, height);
    ellipse(randomNumber(147, 150), y, width, height);
    ellipse(randomNumber(247, 250), y, width, height);
    ellipse(randomNumber(297, 300), y, width, height);
    ellipse(randomNumber(347, 350), y, width, height);
  }
}
```

The abstraction I developed individually is the drawClouds() function which takes in four parameters: x, y, width, and height. This function checks to see if the camera for the game is on and if it is true, the color of the "clouds" is white with a size 3 stroke and the size of the clouds and the y-coordinates are chosen by the y, width, and height parameter declared when the drawClouds() function with four parameters is called inside the draw() function. The x-coordinate for each "cloud" is chosen by a random number, the y parameter is declared as 55, the width is declared as 60, and the height is declared as 50.

This abstraction helps manage the complexity of my program by being able to define the parameters once when the drawClouds() function is being called inside the draw() function instead of needing to type out each parameter individually, especially since each "cloud" has the same y-coordinate, width, and height.