Create — Applications from Ideas
Written Response Submission Template

Submission Requirements

2. Written Responses

Program Purpose and Development

2a.

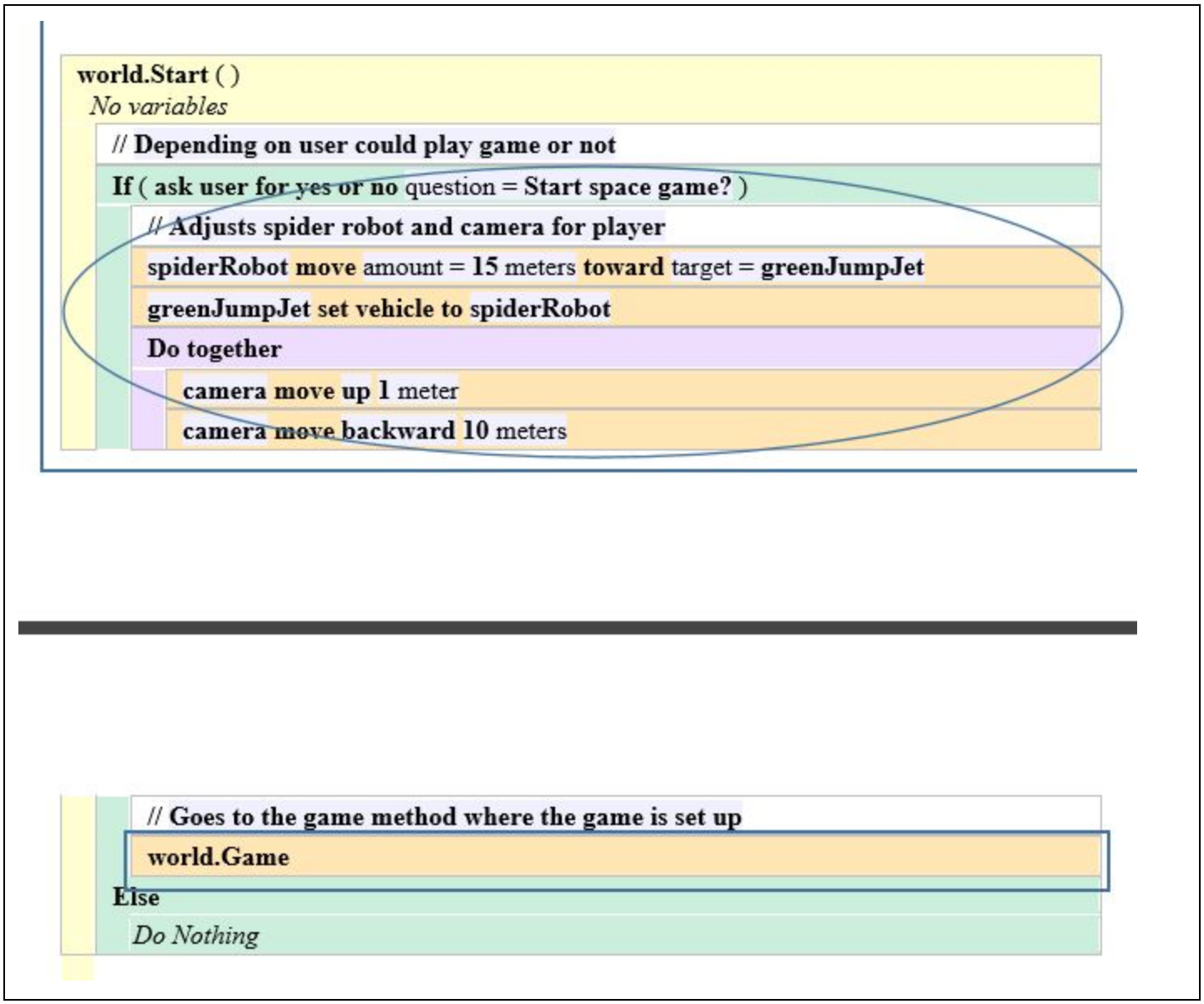
The programming language that I used to create my program was Alice. The purpose of my program is to be an entertainment game and functions by requiring the user to collect 10 orbs using a spaceship to win. If this requirement is not met, the user can try again. The video demonstrates the start of the game by asking the user if they would play or not, if yes then the game would start (if not then the spider robot can just free roam). In the first try, the spaceship collects all orbs and gets launched up into space. The second try, the spaceship does not collect all the orbs in time and the user is asked if they wanted to try again. I clicked no and game stated Game Over but if yes was pressed the game resets. The variables time and points are displayed below the program.

2b.

I independently made the whole program. In the beginning I wanted to make a game about collecting items. First I assembled a list containing of the orbs that will be collected as the points variable. I then assigned arrow controls to the spider controlling the spaceship. I thought about setting it up so when the spaceship is within threshold of an orb, that orb will sink down and 1 point will be added. I also added a point limit that must be met to win.

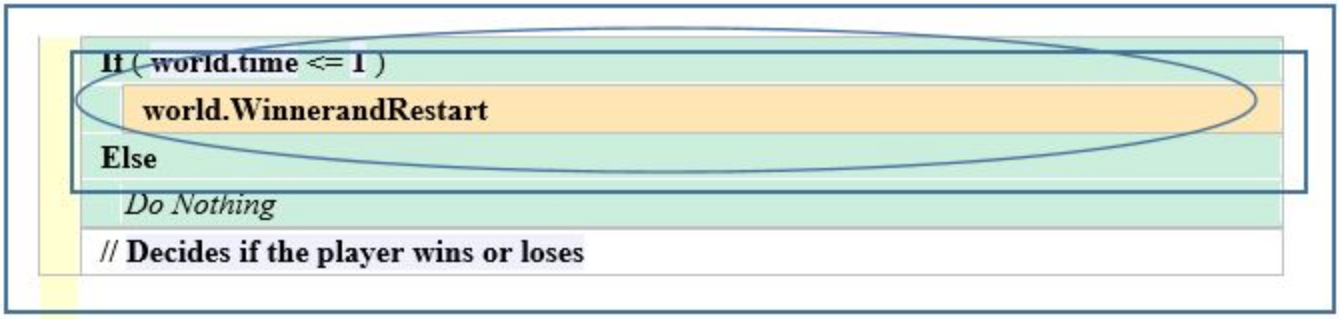
My game was too easy and it wouldn't decide whether a user wins or loses so I took the opportunity to add a timer. I did this by making a time variable and decrementing one off of it every second. The game is decided when the time equals zero. A difficulty I encountered while creating my program was how to restart just the game (not program). Everytime I pressed try again, the whole program freezes. To try to overcome this, I messed around with thee order of code until I solved it when I reset the variables to its original values because when I clicked try again, the conditions were still met so it looped.

2c.


world.Start ( )
No variables
// Depending on user could play game or not
If (ask user for yes or no question = Start space game?)
// Adjusts spider robot and camera for player
spiderRobot move amount = 15 meters toward target = greenJumpJet
greenJumpJet set vehicle to spiderRobot
Do together
camera move up 1 meter
camera move backward 10 meters
// Goes to the game method where the game is set up
world.Game
Else
Do Nothing

```
world.Game ( )
  Orbs = sphere, sphere2, sphere3, sphere4, sphere5, sphere6, sphere7, sphere8, sphere9, sphere10
    // All orbs are listed and commanded to be in a random place each game
    For all Orbs , every item_from_Orbs together
      item_from_Orbs move to greenJumpJet
      item_from_Orbs set color to
      item_from_Orbs move forward ( random number maximum = 75 minimum = -75 )
      item_from_Orbs move left ( random number maximum = 75 minimum = -75 )
    // The main part of the program, tracking the score, time, and orbs
    world.Finder
```

```
world.Finder ( )
  Orbs = sphere, sphere2, sphere3, sphere4, sphere5, sphere7, sphere6, sphere8, sphere9, sphere10
    Do together
      // The timer used in the program
      Loop 40 times times
        decrement world.time by 1 duration = 1 second
        print world.time
      For all Orbs , every item_from_Orbs together
        // Loops the collection of orbbbs so it works for all of them when approached
        While ( world.Collected )
          If ( ( greenJumpJet distance to item_from_Orbs ) < 5 )
            Do together
              item_from_Orbs move down 5 meters
              increment world.coins by 1
              print world.coins
          Else
            Do Nothing
```

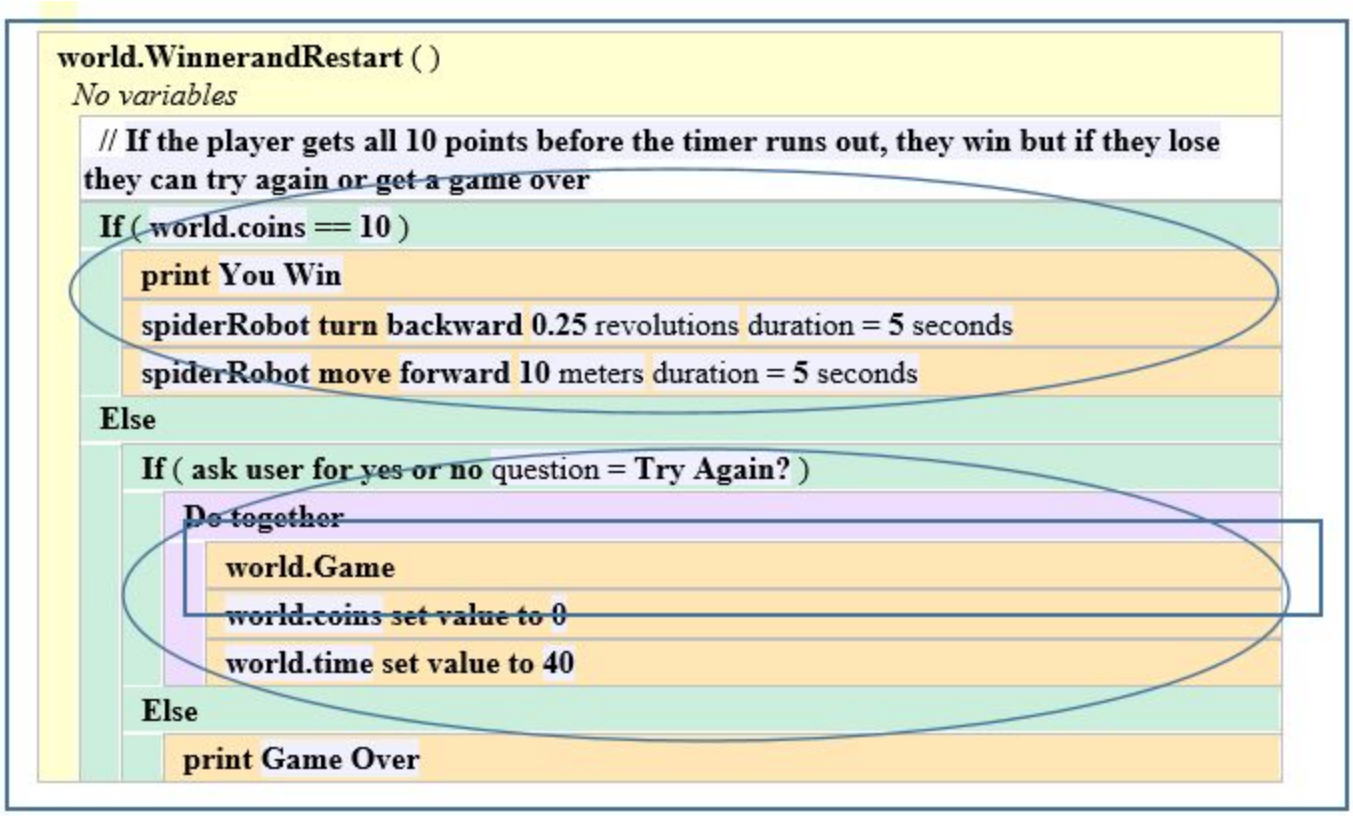


```
If ( world.time <= 1 )
    world.WinnerandRestart
Else
    Do Nothing
// Decides if the player wins or loses
```

My main algorithm world.Start uses sequencing by setting up the spider into space ship. This method uses logic by using an if/else and yes or no function to decide if the player wants the game to start or not. My method world.Start functions by asking a yes or no question about if the user wants to play. Depending on the answer the spider will either be put into the spaceship (Yes) or put into free roam (No). The purpose of world.Start is to be an introduction and a choice for the user if they want to play or not.

The main algorithm includes the two sub algorithms world.Game and world.Finder. One of the sub-algorithms, world.Finder uses math by making it so that when the jet is less than 5 meters from an orb from the orbs list, the orb will add a point and make the orbs disappear. The other sub algorithm, world.Game uses math by using randomizers to set the positions of all the orbs in the orbs list between moving forward 75 to -75 meters and then moving left 75 to -75 meters every game.

2d.



World.WinnerandReset was made by me independently to be able to check and decide whether the player wins or loses. It also gives the user the choice if they lose to either restart the game or accept defeat. I created this method that so that the user can only win if the variable coins equals 10. I created the restart function by replaying the method world.Game method which is the setup of the orbs placement and then restarting the variables.

This abstraction world.WinnerandRestart manages the complexity of the program as it breaks down how the winner is decided. It also makes it so that when the game is restarted the code to make the game work the same way as when it was just started without having to be written again. This method hides all of these complex instructions that make the game replayable. The name also clearly explains the function of the method as it says that the method holds the instruction to decide the winner and to restart the game.