Create — Applications from Ideas
Written Response Submission Template

Submission Requirements

2. Written Responses

Submit one PDF document in which you respond directly to each prompt. Clearly label your responses **2a – 2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

Program Purpose and Development

2a. Identify the programming language and identify the purpose of your program. Explain your video using one of the following:

- A written summary
- of what the video illustrates OR
- An audio narration in your video. If you choose this option, your response to the written summary should read, "The explanation is located in the video."

(Approximately 150 words)

Insert response for 2a in the text box below.

My program is a simple demonstration of a first-person game in a 3D environment. I wrote this program in Python using the IDLE development environment. I also used the Panda3D game engine/library, by Carnegie Mellon Entertainment Technology Center (https://www.panda3d.org/), to manage the camera, player, and environment; to create and render the window and screen; and to manage most all other aspects of a 3D game. The environment demonstrated in the game was also created by Carnegie Mellon Entertainment Technology Center. As a game, it has no objective, however as a program, its purpose is the demonstration of the capabilities of Python concerning a 3D environment. The video demonstrates the basic mechanics of the program—the fact that the player can walk around the screen with the pressing of W, A, S, and D, and that his direction may be controlled through the movement of the mouse—and therefore successfully identifies the key features—and the only features—of the program.

2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development; the second could refer to either collaborative or independent program development. *(Approximately 200 words)*

Insert response for 2b in the text box below.

This program was created independently. One challenge was making the character move based off of both the pressing of "W," "A," "S," and "D," and the angle of the camera. I began by using just the keys—when a key was pressed, the x-position or y-position of the character in the coordinate plane would increase or decrease by a static number, regardless of the camera's angle, resulting in the player moving in one direction regardless of the direction in which he was facing. To solve this, I used the sine and cosine functions with the camera's angle, and added the sines and cosines to the player's x-position and y-position. This resulted in character movements that were accurately based off of the camera's angle. Another difficulty was using the mouse to control the camera angle. I attempted to set the camera's rotation based off of the location of the mouse within the window, however the cursor could be moved off of the window. To solve this, I had the program constantly reset the cursor to the center of said window ('(0, 0)' as far as Panda3D is concerned), and when the mouse was moved away from the center, its new value would be added to the camera's angle before the cursor was reset again.

2c. Capture and paste an image or images of your program code segment that implements the most complex algorithm you wrote. (marked with a **color border** below)

```
    #Method that moves character in-game
    def moveCharacter(self, task):
        #Moving FORWARDS
        if self.keys["w"] == True:

            #90 and 270 are on the wrong sides as far as this operation is cor
            angleForMovement = 360 - self.angleH

            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos += math.cos(math.radians(angleForMovement))
            self.xpos += math.sin(math.radians(angleForMovement))



        #Moving BACKWARDS
        if self.keys["s"] == True:

            #90 and 270 are on the wrong sides as far as this operation is cor
            angleForMovement = 360 - self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine doesn't
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos -= math.cos(math.radians(angleForMovement))
            self.xpos -= math.sin(math.radians(angleForMovement))

        #Moving LEFT
        if self.keys["a"] == True:
            angleForMovement = self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine doesn't
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos -= math.sin(math.radians(angleForMovement))
            self.xpos -= math.cos(math.radians(angleForMovement))


        #Moving RIGHT
        if self.keys["d"] == True:
            angleForMovement = self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine doesn't
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos += math.sin(math.radians(angleForMovement))
            self.xpos += math.cos(math.radians(angleForMovement))

        return Task.cont
```

Your algorithm should integrate several mathematical and logical concepts.
Describe the mathematical and logical concepts used to develop the algorithm.
Explain the complexity of the algorithm and how it functions in the program.
(Approximately 200 words)

Insert text response for 2c in the plain box below.

This algorothm is a method titled 'moveCharacter.' It consists of/contains four smaller algorithms—one for each key that controls the character's movements. Each of the four algorithms is run when its respective key is pressed, hence the 'if' statements managing each of them. In each of the four algorithms that move the character, an if statement will determine whether the camera's angle is 360 degrees, and reset it to 0 if this is the case (another algorithm in use). This is because 0 is equal to 360, as far as rotation is concerned, and the cosine function cannot take 360 as an input. In each of the four algorithms, the sine and cosine of the camera's angle are taken, and applied to the character's x- and y-positions. The variations between the four algorithms are which player position values the sine and cosine are applied to, and whether the sine and cosine values are added or subtracted. These variations account for the four keys moving the player in entirely different directions. Collectively, these four algorithms form a larger algorithm that is responsible for the movement of the character.

2d. Capture and paste an image or images of the program code segment that contains an abstraction you developed (marked with a matching **blue color border** below)

```
    #Method that changes angle of camera
    def lookCharacter(self, task):

        if self.mw.hasMouse():
            mouseX = self.mw.getMouseX()
            mouseY = self.mw.getMouseY()

            #Resets mouse
            base.win.movePointer(0, self.win.getXSize() / 2, self.win.getYSize

            self.angleH += float(mouseX * -30)

            #These account for Windows beign mean with the dimensions of the
            #fullscreen

            #If the window height is an even number, do your thing.
            if float(self.win.getYSize()) % 2.0 == 0.0:

                self.angleP += (float(mouseY) * 30)


            #If the window height is an odd number, which causes a whole load
            #problems do fun things that fix said problems

            elif float(self.win.getYSize()) % 2.0 != 0.0:

                self.angleP += (float(mouseY) - 0.000879507453647) * 30



            #Angle correction
            if self.angleP > 80:
                self.angleP = 80

            if self.angleP < -80:
                self.angleP = -80

            if self.angleH == 360:
                self.angleH = 0

            if self.angleH > 360:
                self.angleH = 360 - self.angleH


        return Task.cont
```

Your abstraction should integrate mathematical and logical concepts.
Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

Insert text response for 2d in the plain box below.

The code above represents abstraction in that a relatively large and complex algorithm is contained in a method, 'lookCharacter,' that is used to simplify and represent the algortithm therein. In the 'init' method of the main class, there is a line of code that adds the method 'lookCharacter' to the list of actions that the game engine Panda3D will enact repeatedly. Each time that Panda3D calls the method 'lookCharacter,' it is referencing a long algorithm through a simple, abstract method name. The abstracted algorithm above will, when called through its amazing, abstract method name, begin by checking whether or not the cursor is in the game window. If it is, it will reset the cursor to the center of the window. Following is a small algorithm that determines whether or not the window height is an odd number—as the cursor cannot be in the direct center of a window with an odd-numbered height—and makes some adjustments if this is the case. It will then ensure that the camera is facing neither too far up nor too far down, and adjusts the camera accordingly.