Client Story:

The subject of this project was selected out of interest. It should be a 3D first-person shooter with multiple levels or areas selected through a menu. There should be several varieties of weapons and enemies, and at least one type of ammunition. There should also be sound effects and music, and the aforementioned menu. The game should receive input through the keyboard and the mouse.

Backlog:

There is an environment with the ability to support the player character and other characters and objects

The player character exists in-game

There are enemy characters

The character may be moved through the use of the keyboard and mouse (Standard WSAD configuration for walking, movement of mouse left and right controls horizontal camera angle, movement of mouse up and down controls vertical angle)

The character has guns

The character may shoot and kill the enemies

Sprint Tasks:  
Load test environment

Create camera

Move camera based off of pressing of W, S, A, and D

Turn camera horizontally and vertically based off of pressing of left and right arrow keys (for testing)

Move camera based off of pressing of W, S, A, and D and camera angle

Turn camera horizontally and vertically based off of mouse movements

Optimize and reorganize code

Make character unable to walk through solid objects

Responses to Questions:

2. Written Responses

Submit one PDF file 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.

2a. Provide a written response or audio narration in your video that:

* identifies the programming language;
* identifies the purpose of your program; and
* explains what the video illustrates.

 (Approximately 150 words)

2.a

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. 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.

(Approximately 200 words)

2.b

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. 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 the program code segment that implements an algorithm (marked with an oval in section 3 below) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Approximately 200 words)

(Approximately 200 words)

2.c

This is a method titled ‘moveCharacter,’ because it moves the character. It is broken up into four 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. 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 algorithm, 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 parts 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.

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 concerned

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 concerned

angleForMovement = 360 - self.angleH

#Make sure angleForMovement is 0 instead of 360, as cosine doesn't work

#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 work

#with 360

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 work

#with 360

if angleForMovement == 360:

angleForMovement = 0

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

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

return Task.cont

2d. Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (Approximately 200 words)

2.d

The code below 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.

def \_\_init\_\_(self):

self.taskMgr.add(self.lookCharacter, "lookCharacter")

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() / 2)

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 of

#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

3. Program Code

Capture and paste your entire program code into the PDF.

Mark with an oval the segment of program code that implements the algorithm you created for your program.

Mark with a rectangle the segment of program code that represents an abstraction you developed.

Include comments or citations for program code that has been written by someone else.

3.

from direct.showbase.ShowBase import ShowBase

from direct.task import Task

from direct.actor.Actor import Actor

from direct.interval.IntervalGlobal import Sequence

from panda3d.core import Point3

from pandac.PandaModules import WindowProperties

import math

import sys

class MyApp(ShowBase):

def \_\_init\_\_(self):

ShowBase.\_\_init\_\_(self)

#Disable mouse controls

self.disableMouse()

#Load enviromnment model

self.scene = self.loader.loadModel("models/environment")

#Reparent model top renderer

self.scene.reparentTo(self.render)

#Apply scale and position transforms on the model

self.scene.setScale(0.25, 0.25, 0.25)

self.scene.setPos(-8, 42, 0)

#Add task spinCameraTask procedure to task manager

self.taskMgr.add(self.setCameraTask, "setCameraTask")

self.taskMgr.add(self.keyInput, "keyInput")

self.taskMgr.add(self.moveCharacter, "movecharacter")

self.taskMgr.add(self.lookCharacter, "lookCharacter")

#Positions of character/camera

self.xpos = 0

self.ypos = 0

self.zpos = 10

self.angleH = 0

self.angleP = 0

self.xInterval = 0.0

self.yInterval = 0.0

#Speed at which the character/camera moves

self.movementInterval = 1

self.lookInterval = 2

#Dictionary of key states

self.keys = {"w" : False,

"s" : False,

"a" : False,

"d" : False,

"arrow\_right" : False,

"arrow\_left": False}

#Configuring cursor settings

wp = WindowProperties()

wp.setMouseMode(WindowProperties.MRelative)

wp.setCursorHidden(True)

self.win.requestProperties(wp)

self.mw = self.mouseWatcherNode

#Define a procedure to move the camera

def setCameraTask(self, task):

self.camera.setPos(self.xpos, self.ypos, self.zpos)

self.camera.setHpr(self.angleH, self.angleP, 0)

return Task.cont

def setWToTrue(self):

self.keys["w"] = True

def setWToFalse(self):

self.keys["w"] = False

def setSToTrue(self):

self.keys["s"] = True

def setSToFalse(self):

self.keys["s"] = False

def setAToTrue(self):

self.keys["a"] = True

def setAToFalse(self):

self.keys["a"] = False

def setDToTrue(self):

self.keys["d"] = True

def setDToFalse(self):

self.keys["d"] = False

def setArrowRightToTrue(self):

self.keys["arrow\_right"] = True

def setArrowRightToFalse(self):

self.keys["arrow\_right"] = False

def setArrowLeftToTrue(self):

self.keys["arrow\_left"] = True

def setArrowLeftToFalse(self):

self.keys["arrow\_left"] = False

def killGame(self):

sys.exitfunc()

sys.exit()

def keyInput(self, task):

self.accept("w", self.setWToTrue)

self.accept("w-up", self.setWToFalse)

self.accept("s", self.setSToTrue)

self.accept("s-up", self.setSToFalse)

self.accept("a", self.setAToTrue)

self.accept("a-up", self.setAToFalse)

self.accept("d", self.setDToTrue)

self.accept("d-up", self.setDToFalse)

self.accept("escape", self.killGame)

return Task.cont

#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() / 2)

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 of

#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

#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 concerned

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 concerned

angleForMovement = 360 - self.angleH

#Make sure angleForMovement is 0 instead of 360, as cosine doesn't work

#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 work

#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 work

#with 0

if angleForMovement == 360:

angleForMovement = 0

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

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

return Task.cont

app = MyApp()

props = WindowProperties()

props.setTitle("Caleb's Game")

app.win.requestProperties(props)

app.run()