2022

K.O Game

Code and assets

5/31/2022

Contents

Code

Main file Code

```
# Python imports
importos

# Panda3D imoprts
fromdirect.showbase.ShowBaseimportShowBase
fromdirect.fsm.FSMimportFSM
fromdirect.gui.DirectGuiimportDGG
from panda3d.core import (
    CollisionTraverser,
    CollisionHandlerPusher,
    AntialiasAttrib,
    ConfigPageManager,
    ConfigVariableInt,
    ConfigVariableBool,
    ConfigVariableString,
    OFileStream,
    WindowProperties,
    loadPrcFileData,
    loadPrcFile,
    Filename,
    AudioSound)
fromdirect.showbase.Audio3DManagerimportAudio3DManager

# Game imports
fromplayerimportPlayer
fromarenaimportArena
frommenuimportMenu
fromcreditsimportCredits
fromcharacterselectionimportCharacterSelection
fromlevelselectionimportLevelSelection
fromkoscreenimportKoScreen
fromhudimportHud
fromhelperimporthide_cursor, show_cursor
# set company and application details
companyName="University Of central Punjab"
appName="K.O"
versionstring="19.07"

# build the path from the details we have
home =os.path.expanduser("~")
basedir=os.path.join(
```

```
    home,
    companyName,
    appName)
ifnotos.path.exists(basedir):
    os.makedirs(basedir)

# look for a config file
prcFile=os.path.join(basedir, "{}.prc".format(appName))
ifos.path.exists(prcFile):
    mainConfig=loadPrcFile(Filename.fromOsSpecific(prcFile))

# set configurations that should not be changed from a config file
loadPrcFileData("",
"""
    #
    # Model loading
    #
    model-path $MAIN_DIR/assets/

    #
    # Window and graphics
    #
    window-title {}
    #show-frame-rate-meter 1

    #
    # Logging
    #
    #notify-level info
    notify-timestamp 1
""".format(appName))

#
# MAIN GAME CLASS
#
classMain(ShowBase, FSM):
    """Main function of the application
    initialise the engine (ShowBase)"""

 def__init__(self):
        """initialise the engine"""
        ShowBase.__init__(self)
        base.notify.info("Version {}".format(versionstring))
        FSM.__init__(self, "FSM-Game")
 # BASIC APPLICATION CONFIGURATIONS
```

```
        # disable pandas default camera driver
        self.disableMouse()
# setantialias for the complete sceen to automatic
        self.render.setAntialias(AntialiasAttrib.MAuto)
# shader generator
        render.setShaderAuto()
        # Enhance font readability
        DGG.getDefaultFont().setPixelsPerUnit(100)
        # get the displays width and height for later usage
        self.dispWidth=self.pipe.getDisplayWidth()
        self.dispHeight=self.pipe.getDisplayHeight()

        #
        # CONFIGURATION LOADING
        #
        # load given variables or set defaults
        # check if particles should be enabled
        # NOTE: If you use the internal physics engine, this always has
        #       to be enabled!
        particles =ConfigVariableBool("particles-enabled", True).getValue()
        if particles:
            self.enableParticles()

        def setFullscreen():
            """Helper function to set the window fullscreen
            with width and height set to the screens size"""
            # set window properties
            # clear all properties not previously set
            base.win.clearRejectedProperties()
            # setup new window properties
            props =WindowProperties()
            # Fullscreen
            props.setFullscreen(True)
            # set the window size to the screen resolution
            props.setSize(self.dispWidth, self.dispHeight)
            # request the new properties
            base.win.requestProperties(props)
            # Set the config variables so we correctly store the
            # new size and fullscreen setting later
            winSize=ConfigVariableString("win-size")
            winSize.setValue("{}{}".format(self.dispWidth, self.dispHeight))
            fullscreen=ConfigVariableBool("fullscreen")
            fullscreen.setValue(True)
            # Render a frame to make sure the fullscreen is applied
            # before we do anything else
```

```
            self.taskMgr.step()
            # make sure to propagate the new aspect ratio properly so
            # the GUI and other things will be scaled appropriately
            aspectRatio=self.dispWidth/self.dispHeight
            self.adjustWindowAspectRatio(aspectRatio)
# check if the config file hasn't been created
        ifnotos.path.exists(prcFile):
            setFullscreen()
        # automatically safe configuration at application exit
        #base.exitFunc = self.__writeConfig

        #
        # INITIALIZE GAME CONTENT
        #
        base.cTrav=CollisionTraverser("base collision traverser")
        base.pusher=CollisionHandlerPusher()
        self.menu=Menu()
        self.credits=Credits()
        self.charSelection=CharacterSelection()
        self.levelSelection=LevelSelection()
        self.koScreen=KoScreen()
        self.hud=Hud()
        self.menuMusic=loader.loadMusic("assets/audio/menuMusic.ogg")
        self.menuMusic.setLoop(True)
        self.fightMusic=loader.loadMusic("assets/audio/fightMusic.ogg")
        self.fightMusic.setLoop(True)
        base.audio3d =Audio3DManager(base.sfxManagerList[0], camera)

        #
        # EVENT HANDLING
        #
        # By default we accept the escape key
        self.accept("escape", self.__escape)

        #
        # ENTER GAMES INITIAL FSM STATE
        #
        self.request("Menu")


    # FSM PART
 defenterMenu(self):
        show_cursor()
        self.accept("Menu-Start", self.request, ["CharSelection"])
        self.accept("Menu-Credits", self.request, ["Credits"])
```

```
        self.accept("Menu-Quit", self.userExit)
        self.ignore("KoScreen-Back")
        self.koScreen.hide()
        self.menu.show()
        ifself.menuMusic.status() !=AudioSound.PLAYING:
            self.menuMusic.play()
        ifself.fightMusic.status() ==AudioSound.PLAYING:
            self.fightMusic.stop()
defexitMenu(self):
        self.ignore("Menu-Start")
        self.ignore("Menu-Credits")
        self.ignore("Menu-Quit")
        self.menu.hide()

    defenterCredits(self):
        self.accept("Credits-Back", self.request, ["Menu"])
        self.koScreen.hide()
        self.credits.show()

    defexitCredits(self):
        self.ignore("Credits-Back")
        self.credits.hide()

    defenterCharSelection(self):
        self.accept("CharSelection-Back", self.request, ["Menu"])
        self.accept("CharSelection-Start", self.request, ["LevelSelection"])
        self.charSelection.show()

    defexitCharSelection(self):
        self.ignore("CharSelection-Start")
        self.ignore("CharSelection-Back")
        self.charSelection.hide()
        self.selectedChar1 =self.charSelection.selectedCharacter1
        self.selectedChar2 =self.charSelection.selectedCharacter2

    defenterLevelSelection(self):
        self.accept("LevelSelection-Back", self.request, ["CharSelection"])
        self.accept("LevelSelection-Start", self.request, ["Game"])
        self.levelSelection.show()

    defexitLevelSelection(self):
        self.ignore("LevelSelection-Start")
        self.ignore("LevelSelection-Back")
        self.levelSelection.hide()
```

```
def enterGame(self):
    # main game code should be called here
    self.arena=Arena(self.levelSelection.selectedLevel)
    self.arena.start()
    self.camera.setPos(0, -5, 1.25)
    self.player=Player(0, self.selectedChar1, "p1")
    self.player2 =Player(1, self.selectedChar2, "p2")
    self.player.setEnemy(self.player2.collisionNodeName)
    self.player2.setEnemy(self.player.collisionNodeName)
    self.player.start(self.arena.getStartPos(1))
    self.player2.start(self.arena.getStartPos(2))
    self.taskMgr.add(self.updateWorldCam, "world camera update task")
    self.accept("gameOver", self.gameOver)
    self.hud.show()
    def lifeChanged(charId, health):
        base.messenger.send(
            "hud_setLifeBarValue",
            [charId, health])
    self.accept("lifeChanged", lifeChanged)
    hide_cursor()
    if self.fightMusic.status() !=AudioSound.PLAYING:
        self.fightMusic.play()
    if self.menuMusic.status() ==AudioSound.PLAYING:
        self.menuMusic.stop()

def exitGame(self):
    # cleanup for game code
    self.taskMgr.remove("world camera update task")
    self.player.stop()
    self.player2.stop()
    del self.player
    del self.player2
    self.arena.stop()
    self.ignore("gameOver")
    self.ignore("lifeChanged")
    self.hud.hide()

#
# FSM PART END
#

#
# BASIC FUNCTIONS
#
def gameOver(self, LoosingCharId):
```

```
    show_cursor()
    winningChar=1
    ifLoosingCharId==0:
        winningChar=2
    self.accept("KoScreen-Back", self.request, ["Credits"])
    self.koScreen.show(winningChar)

def updateWorldCam(self, task):
    playerVec=self.player.getPos() -self.player2.getPos()
    playerDist=playerVec.length()
    x =self.player.getX() +playerDist/2.0
    self.camera.setX(x)

    zoomout=False
    ifnotself.cam.node().isInView(self.player.getPos(self.cam)):
        camPosUpdate=-2*globalClock.getDt()
        self.camera.setY(self.camera, camPosUpdate)
        zoomout=True
    ifnotself.cam.node().isInView(self.player2.getPos(self.cam)):
        camPosUpdate=-2*globalClock.getDt()
        self.camera.setY(self.camera, camPosUpdate)
        zoomout=True
    ifnotzoomout:
        ifself.camera.getY() <-5:
            camPosUpdate=2*globalClock.getDt()
            self.camera.setY(self.camera, camPosUpdate)
    returntask.cont

def __escape(self):
    """Handle user escape key klicks"""
    ifself.state=="Menu":
        # In this state, we will stop the application
        self.userExit()
    elifself.state=="LevelSelection":
        self.request("CharSelection")
    else:
        # In every other state, we switch back to the Menu state
        self.request("Menu")

def __writeConfig(self):
    """Save current config in the prc file or if no prc file exists
    create one. The prc file is set in the prcFile variable"""
    page =None

    #
```

```
        #TODO: add any configuration variable names that you have added
        #      to the dictionaries in the next lines. Set the current
        #      configurations value as value in this dictionary and it's
        #      name as key.
        configVariables= {
            # set the window size in the config file
            "win-size": ConfigVariableString("win-size",
"{}{}".format(self.dispWidth, self.dispHeight)).getValue(),
            # set the default to fullscreen in the config file
            "fullscreen": "#t"ifConfigVariableBool("fullscreen", True).getValue()
else"#f",
            # particles
            "particles-enabled": "#t"ifself.particleMgrEnabledelse"#f",
            # audio
            "audio-volume": str(round(self.musicManager.getVolume(), 2)),
            "audio-music-active": "#t"ifConfigVariableBool("audio-music-
active").getValue() else"#f",
            "audio-sfx-active": "#t"ifConfigVariableBool("audio-sfx-
active").getValue() else"#f",
            # logging
            "notify-output": os.path.join(basedir, "game.log"),
            # window
            "framebuffer-multisample": "#t"ifConfigVariableBool("framebuffer-
multisample").getValue() else"#f",
            "multisamples": str(ConfigVariableInt("multisamples", 8).getValue()),
            "texture-anisotropic-degree": str(ConfigVariableInt("texture-
anisotropic-degree").getValue()),
            "textures-auto-power-2": "#t"ifConfigVariableBool("textures-auto-
power-2", True).getValue() else"#f",
            }

        page =None
        # Check if we have an existing configuration file
        ifos.path.exists(prcFile):
            # open the config file and change values according to current
            # application settings
            page =loadPrcFile(Filename.fromOsSpecific(prcFile))
            removeDecls= []
            for dec inrange(page.getNumDeclarations()):
                # Check if our variables are given.
                # NOTE: This check has to be done to not loose our base or other
                #       manual config changes by the user
                ifpage.getVariableName(dec) inconfigVariables.keys():
                    removeDecls.append(page.modifyDeclaration(dec))
            for dec inremoveDecls:
```

```
                page.deleteDeclaration(dec)
        else:
            # Create a config file and set default values
            cpMgr=ConfigPageManager.getGlobalPtr()
            page =cpMgr.makeExplicitPage("Application Config")

        # always write custom configurations
        for key, value inconfigVariables.items():
            page.makeDeclaration(key, value)
        # create a stream to the specified config file
        configfile=OFileStream(prcFile)
        # and now write it out
        page.write(configfile)
        # close the stream
        configfile.close()

    #
    # BASIC END
    #
# CLASS Main END

#
# START GAME
#
Game =Main()
Game.run()
```

Menu File Code

```
from panda3d.core importTextNode
fromdirect.gui.DirectGuiimport (
    DirectFrame,
    DirectLabel,
    DirectButton)
classMenu:
    def__init__(self):

        self.frameMain=DirectFrame(
            image="gui/MenuBackground.png",
            image_scale= (1.7778, 1, 1),
            frameSize= (base.a2dLeft, base.a2dRight,
                        base.a2dBottom, base.a2dTop),
            frameColor= (0, 0, 0, 0))
        self.frameMain.setTransparency(1)

        self.title=DirectLabel(
            scale=0.15,
            text_align=TextNode.ALeft,
            pos= (base.a2dLeft +0.2, 0, 0),
            frameColor= (0, 0, 0, 0),
            text="Main Menu",
            text_fg= (1,1,1,1))
        self.title.setTransparency(1)
        self.title.reparentTo(self.frameMain)

        self.btnStart=self.createButton(
            "Start",
            -.10,
            ["Menu-Start"])

        self.btnStart=self.createButton(
            "Credits",
            -.25,
            ["Menu-Credits"])

        self.btnExit=self.createButton(
           "Quit",
            -.40,
            ["Menu-Quit"])

        self.hide()

    defcreateButton(self, text, verticalPos, eventArgs):
```

```
        maps =loader.loadModel("gui/button_map")
        btnGeom= (maps.find("**/btn_ready"),
                  maps.find("**/btn_click"),
                  maps.find("**/btn_rollover"),
                  maps.find("**/btn_disabled"))
        btn=DirectButton(
            text=text,
            text_fg= (0,0,0,1),
            text_scale=0.05,
            text_pos= (0.02, -0.015),
            text_align=TextNode.ALeft,
            scale=2,
            pos= (base.a2dLeft +0.2, 0, verticalPos),
            geom=btnGeom,
            relief=0,
            frameColor= (0,0,0,0),
            command=base.messenger.send,
            extraArgs=eventArgs,
            pressEffect=False,
            rolloverSound=None,
            clickSound=None)
        btn.reparentTo(self.frameMain)

    defshow(self):
        self.frameMain.show()

    defhide(self):
        self.frameMain.hide()
```

Arena File Code

```
from panda3d.core import (
    AmbientLight,
    PerspectiveLens,
    DirectionalLight,
    Fog)

fromdirect.particles.ParticleEffectimportParticleEffect

classArena:
    def__init__(self, arenaNr):
        arenaPath="levels/arena{}/".format(arenaNr)
        self.arena=loader.loadModel(arenaPath+"arena")
        self.arena.setScale(2)
        self.arena.reparentTo(render)
        self.arena.hide()

        ambientLight=AmbientLight("ambient_light")
        ambientLight.setColor((0.2, 0.2, 0.2, 1))
        self.alnp=render.attachNewNode(ambientLight)

        sunLens=PerspectiveLens()
        sunLens.setFilmSize(50)
        sun =DirectionalLight("sun")
        sun.setColor((1, 1, 1, 1))
        sun.setShadowCaster(True, 2048, 2048)
        sun.setScene(render)
        #sun.showFrustum()

        self.ambientSound=None
        self.levelParticles=None
        ifarenaNr==1:
            sunLens.setNearFar(25,45)
            sun.setLens(sunLens)
            self.sunNp=render.attachNewNode(sun)
            self.sunNp.setPos(-10, -10, 30)
            self.sunNp.lookAt(0,0,0)

            self.ambientSound=loader.loadSfx("assets/audio/ambientLevel1.ogg")
            self.ambientSound.setLoop(True)

            self.fog=Fog("Outside Fog")
            self.fog.setColor(0.3,0.3,0.5)
            self.fog.setExpDensity(0.025)
```

```
            self.levelParticles=ParticleEffect()
            self.levelParticles.loadConfig("assets/fx/Leafs.ptf")
            self.levelParticles.start(parent= render2d, renderParent= render2d)
        elifarenaNr==2:
            sunLens.setFov(120, 40)
            sunLens.setNearFar(2,10)
            sun.setLens(sunLens)
            self.sunNp=render.attachNewNode(sun)
            self.sunNp.setPos(0, 0, 5)
            self.sunNp.lookAt(0,0,0)

            self.fog=Fog("Temple Fog")
            self.fog.setColor(0,0,0)
            self.fog.setExpDensity(0.065)

    defstart(self):
        self.arena.show()
        render.setLight(self.alnp)
        render.setLight(self.sunNp)
        ifself.ambientSound!=None:
            self.ambientSound.play()
        render.setFog(self.fog)

    defstop(self):
        self.arena.hide()
        render.clearLight()
        ifself.ambientSound!=None:
            self.ambientSound.stop()
        render.clearFog()
        ifself.levelParticles!=None:
            self.levelParticles.cleanup()

    defgetStartPos(self, charNr):
        ifcharNr==1:
            returnself.arena.find("**/StartPosA").getPos() *2
        elifcharNr==2:
            returnself.arena.find("**/StartPosB").getPos() *2
        else:
            return (0,0,0)
```

Player File Code

```
# Panda3D imoprts
fromdirect.showbase.DirectObjectimportDirectObject
fromdirect.actor.ActorimportActor
fromdirect.fsm.FSMimportFSM
from panda3d.core import (
    CollisionSegment,
    CollisionSphere,
    CollisionNode,
    KeyboardButton,
    AudioSound)
fromdirect.particles.ParticleEffectimportParticleEffect
classPlayer(FSM, DirectObject):
    def__init__(self, charId, charNr, controls):
        FSM.__init__(self, "FSM-Player{}".format(charNr))
        self.charId=charId
        charPath="characters/character{}/".format(charNr)
        self.character=Actor(
            charPath+"char", {
                "Idle":charPath+"idle",
                "Walk":charPath+"walk",
                "Walk_back":charPath+"walk_back",
                "Punch_l":charPath+"punch_l",
                "Punch_r":charPath+"punch_r",
                "Kick_l":charPath+"kick_l",
                "Kick_r":charPath+"kick_r",
                "Defend":charPath+"defend",
                "Hit":charPath+"hit",
                "Defeated":charPath+"defeated"
            }
        )
        self.character.reparentTo(render)
        self.character.hide()
        self.walkSpeed=2.0# units per second
```

```
    ifcontrols=="p1":
        self.character.setH(90)
        self.leftButton=KeyboardButton.asciiKey(b"d")
        self.rightButton=KeyboardButton.asciiKey(b"f")
        self.punchLButton=KeyboardButton.asciiKey(b"q")
        self.punchRButton=KeyboardButton.asciiKey(b"w")
        self.kickLButton=KeyboardButton.asciiKey(b"a")
        self.kickRButton=KeyboardButton.asciiKey(b"s")
        self.defendButton=KeyboardButton.asciiKey(b"e")
    elifcontrols=="p2":
        self.character.setH(-90)
        self.leftButton=KeyboardButton.right()
        self.rightButton=KeyboardButton.left()
        self.punchLButton=KeyboardButton.asciiKey(b"i")
        self.punchRButton=KeyboardButton.asciiKey(b"o")
        self.kickLButton=KeyboardButton.asciiKey(b"k")
        self.kickRButton=KeyboardButton.asciiKey(b"l")
        self.defendButton=KeyboardButton.asciiKey(b"p")

    self.getPos=self.character.getPos
    self.getX=self.character.getX

    characterSphere=CollisionSphere(0, 0, 1.0, 0.5)
    self.collisionNodeName="character{}Collision".format(charId)
    characterColNode=CollisionNode(self.collisionNodeName)
    characterColNode.addSolid(characterSphere)
    self.characterCollision=self.character.attachNewNode(characterColNode)
    # Uncomment this line to show collision solids
    #self.characterCollision.show()
    base.pusher.addCollider(self.characterCollision, self.character)
    base.cTrav.addCollider(self.characterCollision, base.pusher)

    characterHitRay=CollisionSegment(0, -0.5, 1.0, 0, -0.8, 1.0)
    characterColNode.addSolid(characterHitRay)

    self.audioStep= base.audio3d.loadSfx("assets/audio/step.ogg")
    self.audioStep.setLoop(True)
    base.audio3d.attachSoundToObject(self.audioStep, self.character)

    self.audioHit= base.audio3d.loadSfx("assets/audio/hit.ogg")
    self.audioHit.setLoop(False)
    base.audio3d.attachSoundToObject(self.audioStep, self.character)

defsetEnemy(self, enemyColName):
    self.enemyColName=enemyColName
```

```
    inEvent="{}-into-{}".format(enemyColName,self.collisionNodeName)
    base.pusher.addInPattern(inEvent)
    self.accept(inEvent, self.setCanBeHit, [True])
    outEvent="{}-out-{}".format(enemyColName,self.collisionNodeName)
    base.pusher.addOutPattern(outEvent)
    self.accept(outEvent, self.setCanBeHit, [False])

def setCanBeHit(self, yes, collission):
    eventName="hitEnemy{}".format(self.collisionNodeName)
    if yes:
        self.accept(eventName, self.gotHit)
    else:
        self.ignore(eventName)
    self.canBeHit=yes

def gotHit(self):
    if not self.canBeHit or self.isDefending: return

    self.bloodsplat=ParticleEffect()
    self.bloodsplat.loadConfig("assets/fx/BloodSplat.ptf")
    floater =self.character.attachNewNode("particleFloater")
    if self.character.getH() ==90:
        floater.setPos(-1, 0, 1)
    if self.character.getH() ==-90:
        floater.setPos(1, 0, 1)
    self.bloodsplat.start(parent= floater, renderParent= render)
    taskMgr.doMethodLater(0.5, self.bloodsplat.cleanup,
        "stop Particle", extraArgs= [])

    self.health-=10
    base.messenger.send(
        "lifeChanged",
        [self.charId, self.health])
    if self.health<=0:
        self.gotDefeated=True
        self.request("Defeated")
        base.messenger.send("gameOver", [self.charId])
    else:
        self.request("Hit")

def attackAnimationPlaying(self):
    actionAnimations= [
        "Punch_l",
        "Punch_r",
        "Kick_l",
```

```
        "Kick_r",
        "Hit"]
    ifself.character.getCurrentAnim() inactionAnimations: returnTrue

def start(self, startPos):
    self.character.setPos(startPos)
    self.character.show()
    self.request("Idle")
    self.canBeHit=False
    self.isDefending=False
    self.gotDefeated=False
    self.health=100
    taskMgr.add(self.moveTask, "move task {}".format(self.charId))

def stop(self):
    taskMgr.remove("move task {}".format(self.charId))
    self.ignoreAll()
    base.audio3d.detachSound(self.audioStep)
    base.audio3d.detachSound(self.audioHit)
    self.character.cleanup()
    self.character.removeNode()

def moveTask(self, task):
    ifself.gotDefeated:
        base.messenger.send("GameOver")
        returntask.done
    ifself.attackAnimationPlaying(): returntask.cont
    speed =0.0
    isDown=base.mouseWatcherNode.isButtonDown

    ifisDown(self.defendButton):
        ifself.state!="Defend":
            self.isDefending=True
            self.request("Defend")
        returntask.cont
    self.isDefending=False

    # Check for attack keys
    isAction=False
    ifisDown(self.punchLButton):
        isAction=True
        self.request("Punch_l")
    elifisDown(self.punchRButton):
        isAction=True
        self.request("Punch_r")
```

```
    elifisDown(self.kickLButton):
        isAction=True
        self.request("Kick_l")
    elifisDown(self.kickRButton):
        isAction=True
        self.request("Kick_r")
    ifisAction:
        base.messenger.send("hitEnemy{}".format(self.enemyColName))
        returntask.cont

    ifisDown(self.leftButton):
        speed +=self.walkSpeed
    ifisDown(self.rightButton):
        speed -=self.walkSpeed
    yDelta= speed *globalClock.getDt()
    self.character.setY(self.character, yDelta)
    ifspeed !=0.0andself.state!="Walk"andself.state!="Walk_back":
        if speed <0:
            self.request("Walk")
        else:
            self.request("Walk_back")
    elif speed ==0.0andself.state!="Idle":
        self.request("Idle")
    returntask.cont

defenterIdle(self):
    self.character.loop("Idle")
defexitIdle(self):
    self.character.stop()

defenterWalk(self):
    self.character.loop("Walk")
    ifself.audioStep.status() !=AudioSound.PLAYING:
        self.audioStep.play()
defexitWalk(self):
    self.character.stop()
    ifself.audioStep.status() ==AudioSound.PLAYING:
        self.audioStep.stop()

defenterWalk_back(self):
    self.character.loop("Walk_back")
    ifself.audioStep.status() !=AudioSound.PLAYING:
        self.audioStep.play()
defexitWalk_back(self):
    self.character.stop()
```

```
    ifself.audioStep.status() ==AudioSound.PLAYING:
        self.audioStep.stop()

defenterPunch_l(self):
    self.character.play("Punch_l")
defexitPunch_l(self):
    self.character.stop()

defenterPunch_r(self):
    self.character.play("Punch_r")
defexitPunch_r(self):
    self.character.stop()

defenterKick_l(self):
    self.character.play("Kick_l")
defexitKick_l(self):
    self.character.stop()

defenterKick_r(self):
    self.character.play("Kick_r")
defexitKick_r(self):
    self.character.stop()

defenterDefend(self):
    self.character.play("Defend")
defexitDefend(self):
    self.character.stop()

defenterHit(self):
    self.character.play("Hit")
    self.audioHit.play()
defexitHit(self):
    self.character.stop()

defenterDefeated(self):
    self.character.play("Defeated")
defexitDefeated(self):
    self.character.stop()
```

Player Selection File Code

```
from panda3d.core import (
    TextNode,
    Texture)
fromdirect.gui.DirectGuiimport (
    DirectFrame,
    DirectButton,
    DGG)

classCharacterSelection:
    def__init__(self):

        self.frameMain=DirectFrame(
            frameSize= (base.a2dLeft, base.a2dRight,
                        base.a2dBottom, base.a2dTop),
            frameColor= (0.05, 0.05, 0.05, 1))
        self.frameMain.setTransparency(1)

        width =abs(base.a2dLeft) + base.a2dRight

        red =loader.loadTexture("assets/gui/CharRedBG.png")
        red.setWrapU(Texture.WM_repeat)
        red.setWrapV(Texture.WM_repeat)
        self.char1Frame =DirectFrame(
            text="Player 1",
            text_fg= (1,1,1,1),
            text_scale=0.1,
            text_pos= (0, base.a2dTop -0.2),
            frameSize= (-width/6.0, width/6.0,
                        base.a2dBottom, base.a2dTop),
            frameTexture= red,
            pos= (base.a2dLeft+width/6.0, 0, 0))
        self.char1Frame.updateFrameStyle()
        self.char1Frame.setTransparency(1)
        self.char1Frame.reparentTo(self.frameMain)

        blue =loader.loadTexture("assets/gui/CharBlueBG.png")
        blue.setWrapU(Texture.WM_repeat)
        blue.setWrapV(Texture.WM_repeat)
        self.char2Frame =DirectFrame(
            text="Player 2",
            text_fg= (1,1,1,1),
            text_scale=0.1,
            text_pos= (0, base.a2dTop -0.2),
            frameSize= (-width/6.0, width/6.0,
```

```
                base.a2dBottom, base.a2dTop),
    frameTexture= blue,
    pos= (base.a2dRight-width/6.0, 0, 0))
self.char2Frame.setTransparency(1)
self.char2Frame.reparentTo(self.frameMain)

self.footerFrame=DirectFrame(
    text="PLAYER 1 - CHOOSE YOUR CHARACTER",
    text_fg= (1,1,1,1),
    text_scale=0.08,
    text_pos= (0, -0.03),
    frameSize= (base.a2dLeft, base.a2dRight,
                0.1, -0.1),
    pos= (0, 0, base.a2dBottom +0.2),
    frameColor= (0, 0, 0, 0.5))
self.footerFrame.setTransparency(1)
self.footerFrame.reparentTo(self.frameMain)

self.charSelectFrame=DirectFrame(
    text="VS",
    text_fg= (1,1,1,1),
    text_scale=0.1,
    text_pos= (0, base.a2dTop -0.2),
    frameSize= (-width/6.0, width/6.0,
                base.a2dBottom, base.a2dTop),
    frameColor= (0,0,0,0))
self.charSelectFrame.reparentTo(self.frameMain)

self.btnChar1 =self.createCharacterButton(
    (-0.2, 0, 0),
    "assets/gui/Char1Button.png",
    1)
self.btnChar1.reparentTo(self.charSelectFrame)

self.btnChar2 =self.createCharacterButton(
    (0.2, 0, 0),
    "assets/gui/Char2Button.png",
    2)
self.btnChar2.reparentTo(self.charSelectFrame)

self.btnBack=DirectButton(
    text="BACK",
    text_fg= (1,1,1,1),
    text_align=TextNode.ALeft,
    scale=0.1,
```

```
            pad= (0.15, 0.15),
            pos= (base.a2dLeft +0.08, 0, -0.03),
            frameColor= (
                (0.2,0.2,0.2,0.8),
                (0.4,0.4,0.4,0.8),
                (0.4,0.4,0.4,0.8),
                (0.1,0.1,0.1,0.8)),
            relief=1,
            command=base.messenger.send,
            extraArgs= ["CharSelection-Back"],
            pressEffect=False,
            rolloverSound=None,
            clickSound=None)
        self.btnBack.setTransparency(1)
        self.btnBack.reparentTo(self.footerFrame)

        self.btnStart=DirectButton(
            text="START",
            text_fg= (1,1,1,1),
            text_align=TextNode.ARight,
            scale=0.1,
            pad= (0.15, 0.15),
            pos= (base.a2dRight -0.08, 0, -0.03),
            relief=1,
            frameColor= (
                (0.2,0.2,0.2,0.8),
                (0.4,0.4,0.4,0.8),
                (0.4,0.4,0.4,0.8),
                (0.1,0.1,0.1,0.8)),
            command=base.messenger.send,
            extraArgs= ["CharSelection-Start"],
            pressEffect=False,
            rolloverSound=None,
            clickSound=None)
        self.btnStart.setTransparency(1)
        self.btnStart.reparentTo(self.footerFrame)
        self.btnStart["state"] =DGG.DISABLED

        self.hide()

    defcreateCharacterButton(self, pos, image, charNr):
        btn=DirectButton(
            scale=0.1,
            relief=0,
            frameColor= (0,0,0,0),
```

```
            pos=pos,
            image=image,
            command=self.selectCharacter,
            extraArgs= [charNr],
            rolloverSound=None,
            clickSound=None)
        btn.setTransparency(1)
        returnbtn

    defselectCharacter(self, charNr):
        ifself.char1Frame["image"] ==None:
            self.char1Frame["image"] ="assets/gui/Char{}_L.png".format(charNr)
            self.char1Frame["image_scale"] = (0.5,1, 1)
            self.selectedCharacter1 =charNr
            self.footerFrame["text"] ="PLAYER 2 - CHOOSE YOUR CHARACTER"
        elifself.char2Frame["image"] ==None:
            self.char2Frame["image"] ="assets/gui/Char{}_R.png".format(charNr)
            self.char2Frame["image_scale"] = (0.5,1, 1)
            self.selectedCharacter2 =charNr
            self.btnStart["state"] =DGG.NORMAL
            self.footerFrame["text"] ="START THE FIGHT >"

    defshow(self):
        self.selectedCharacter1 =None
        self.selectedCharacter2 =None
        self.char1Frame["image"] =None
        self.char2Frame["image"] =None
        self.footerFrame["text"] ="PLAYER 1 - CHOOSE YOUR CHARACTER"
        self.btnStart["state"] =DGG.DISABLED
        self.frameMain.show()

    defhide(self):
        self.frameMain.hide()
```

Level Selection File Code

```
from panda3d.core importTextNode
fromdirect.gui.DirectGuiimport (
    DirectFrame,
    DirectButton)
classLevelSelection:
    def__init__(self):

        self.frameMain=DirectFrame(
            frameSize= (base.a2dLeft, base.a2dRight,
                        base.a2dBottom, base.a2dTop),
            frameColor= (0.05, 0.05, 0.05, 1))
        self.frameMain.setTransparency(1)

        self.btnLevel1 =self.createLevelButton(
            (-0.6, 0, 0.15),
            "assets/gui/Level1Button.png",
            1)
        self.btnLevel1.reparentTo(self.frameMain)

        self.btnLevel2 =self.createLevelButton(
            (0.6, 0, 0.15),
            "assets/gui/Level2Button.png",
            2)
        self.btnLevel2.reparentTo(self.frameMain)

        self.footerFrame=DirectFrame(
            text="SELECT THE ARENA",
            text_fg= (1,1,1,1),
            text_scale=0.08,
            text_pos= (0, -0.03),
            frameSize= (base.a2dLeft, base.a2dRight,
                        0.1, -0.1),
            pos= (0, 0, base.a2dBottom +0.2),
            frameColor= (0, 0, 0, 0.5))
        self.footerFrame.setTransparency(1)
        self.footerFrame.reparentTo(self.frameMain)

        self.btnBack=DirectButton(
            text="BACK",
            text_fg= (1,1,1,1),
            text_align=TextNode.ALeft,
            scale=0.1,
            pad= (0.15, 0.15),
            pos= (base.a2dLeft +0.08, 0, -0.03),
```

```
        frameColor= (
            (0.2,0.2,0.2,0.8),
            (0.4,0.4,0.4,0.8),
            (0.4,0.4,0.4,0.8),
            (0.1,0.1,0.1,0.8),
            ),
        relief=1,
        command=base.messenger.send,
        extraArgs= ["LevelSelection-Back"],
        pressEffect=False,
        rolloverSound=None,
        clickSound=None)
    self.btnBack.setTransparency(1)
    self.btnBack.reparentTo(self.footerFrame)

    self.hide()

def createLevelButton(self, pos, image, levelNr):
    btn=DirectButton(
        scale= (0.5, 1, 0.75),
        relief=0,
        frameColor= (0,0,0,0),
        pos=pos,
        image=image,
        command=self.selectLevel,
        extraArgs= [levelNr],
        rolloverSound=None,
        clickSound=None)
    btn.setTransparency(1)
    returnbtn

def selectLevel(self, level):
    self.selectedLevel=level
    base.messenger.send("LevelSelection-Start")

def show(self):
    self.frameMain.show()

def hide(self):
    self.frameMain.hide()
```

KOscreen File Code

```
# Panda3D imoprts
fromdirect.showbase.DirectObjectimportDirectObject
fromdirect.gui.DirectGuiimport (
    DirectFrame,
    DirectLabel,
    DirectButton)
classKoScreen(DirectObject):
    def__init__(self):
        self.frameMain=DirectFrame(
            frameSize= (base.a2dLeft, base.a2dRight,
                        base.a2dBottom, base.a2dTop),
            frameColor= (0, 0, 0, 0.75))
        self.frameMain.setTransparency(1)

        self.lbl_KO=DirectLabel(
            text="K.O.",
            text_fg= (1,1,1,1),
            scale=1,
            pos= (0, 0, 0),
            frameColor= (0,0,0,0))
        self.lbl_KO.setTransparency(1)
        self.lbl_KO.reparentTo(self.frameMain)

        self.lbl_PlayerXWon=DirectLabel(
            text="PLAYER X WON",
            text_fg= (1,1,1,1),
            scale=0.25,
            pos= (0, 0, -0.5),
            frameColor= (0,0,0,0))
        self.lbl_PlayerXWon.setTransparency(1)
        self.lbl_PlayerXWon.reparentTo(self.frameMain)

        self.btnContinue=DirectButton(
            text="CONTINUE",
            text_fg= (1,1,1,1),
            scale=0.1,
            pad= (0.15, 0.15),
            pos= (0, 0, -0.8),
            frameColor= (
                (0.2,0.2,0.2,0.8),
                (0.4,0.4,0.4,0.8),
                (0.4,0.4,0.4,0.8),
                (0.1,0.1,0.1,0.8),
                ),
```

```
            relief=1,
            command=base.messenger.send,
            extraArgs= ["KoScreen-Back"],
            pressEffect=False,
            rolloverSound=None,
            clickSound=None)
        self.btnContinue.setTransparency(1)
        self.btnContinue.reparentTo(self.frameMain)

        self.hide()

    defshow(self, succseedingPlayer):
        self.frameMain.show()
        self.lbl_PlayerXWon["text"] ="PLAYER {}WON".format(succseedingPlayer)

    defhide(self):
        self.frameMain.hide()
```

Hud File Code

```
# Panda3D imoprts
fromdirect.showbase.DirectObjectimportDirectObject
fromdirect.gui.DirectGuiimportDirectWaitBar, DGG
from panda3d.core importTextNode

classHud(DirectObject):
    def__init__(self):
        self.lifeBar1 =DirectWaitBar(
            text="Player1",
            text_fg= (1,1,1,1),
            text_pos= (-1.2, -0.18, 0),
            text_align=TextNode.ALeft,
            value=100,
            barColor= (0, 1, 0.25, 1),
            barRelief=DGG.RAISED,
            barBorderWidth= (0.03, 0.03),
            borderWidth= (0.01, 0.01),
            relief=DGG.RIDGE,
            frameColor= (0.8,0.05,0.10,1),
            frameSize= (-1.2, 0, 0, -0.1),
            pos= (-0.2,0,base.a2dTop-0.15))
        self.lifeBar1.setTransparency(1)

        self.lifeBar2 =DirectWaitBar(
            text="Player2",
            text_fg= (1,1,1,1),
            text_pos= (1.2, -0.18, 0),
            text_align=TextNode.ARight,
            value=100,
            barColor= (0, 1, 0.25, 1),
            barRelief=DGG.RAISED,
            barBorderWidth= (0.03, 0.03),
            borderWidth= (0.01, 0.01),
            relief=DGG.RIDGE,
            frameColor= (0.8,0.05,0.10,1),
            frameSize= (0, 1.2, 0, -0.1),
            pos= (0.2,0,base.a2dTop-0.15))
        self.lifeBar2.setTransparency(1)

        self.accept("hud_setLifeBarValue", self.setLifeBarValue)
        self.hide()

    defshow(self):
        self.lifeBar1["value"] =100
```

```
        self.lifeBar2["value"] =100
        self.lifeBar1.show()
        self.lifeBar2.show()

    defhide(self):
        self.lifeBar1.hide()
        self.lifeBar2.hide()

    defsetLifeBarValue(self, barNr, newValue):
        ifbarNr==0:
            self.lifeBar1["value"] =newValue
        elifbarNr==1:
            self.lifeBar2["value"] =newValue
```

helper File code

```
# PYTHON IMPORTS
#
importsys

#
# PANDA3D ENGINE IMPORTS
#
from panda3d.core importWindowProperties

defhide_cursor():
    """set the Cursor invisible"""
    props =WindowProperties()
    props.setCursorHidden(True)
    # somehow the window gets undecorated after hiding the cursor
    # so we reset it here to the value we need
    #props.setUndecorated(settings.fullscreen)
    base.win.requestProperties(props)

defshow_cursor():
    """set the Cursor visible again"""
    props =WindowProperties()
    props.setCursorHidden(False)
    # set the filename to the mouse cursor
    x11 ="assets/gui/Cursor.x11"
    win ="assets/gui/Cursor.pico"
    ifsys.platform.startswith("linux"):
        props.setCursorFilename(x11)
    else:
        props.setCursorFilename(win)
    base.win.requestProperties(props)
```

Credits File Code

```
from panda3d.core import (
    TextNode,
    TextProperties,
    TextPropertiesManager)
fromdirect.gui.DirectGuiimport (
    DirectFrame,
    DirectLabel,
    DirectButton)
fromdirect.stdpy.fileimportopen
fromdirect.interval.LerpIntervalimportLerpPosInterval
classCredits:
    def__init__(self):

        self.frameMain=DirectFrame(
            frameSize= (base.a2dLeft, base.a2dRight,
                        base.a2dBottom, base.a2dTop),
            frameColor= (0.05, 0.05, 0.05, 1))
        self.frameMain.setTransparency(1)

        tpBig=TextProperties()
        tpBig.setTextScale(1.5)
        tpSmall=TextProperties()
        tpSmall.setTextScale(0.75)
        tpUs=TextProperties()
        tpUs.setUnderscore(True)
        tpMgr=TextPropertiesManager.getGlobalPtr()
        tpMgr.setProperties("big", tpBig)
        tpMgr.setProperties("small", tpSmall)
        tpMgr.setProperties("us", tpUs)

        creditsText=""
        withopen("credits.txt") as f:
            creditsText=f.read()
        self.lblCredits=DirectLabel(
            text=creditsText,
            text_fg= (1,1,1,1),
            text_bg= (0,0,0,0),
            frameColor= (0,0,0,0),
            text_align=TextNode.ACenter,
            scale=0.1,
            pos= (0, 0, base.a2dTop -0.2))
        self.lblCredits.setTransparency(1)
        self.lblCredits.reparentTo(self.frameMain)
```

```
        self.creditsScroll=LerpPosInterval(
            self.lblCredits,
            12.0,
            (0, 0, base.a2dTop +3.5),
            startPos=(0, 0, base.a2dBottom),
            name="CreditsScroll")

        self.btnBack=DirectButton(
            text="BACK",
            text_fg= (1,1,1,1),
            text_align=TextNode.ALeft,
            scale=0.1,
            pad= (0.15, 0.15),
            pos= (base.a2dLeft +0.08, 0, base.a2dBottom +0.05),
            frameColor= (
                (0.2,0.2,0.2,0.8),
                (0.4,0.4,0.4,0.8),
                (0.4,0.4,0.4,0.8),
                (0.1,0.1,0.1,0.8),
                ),
            relief=1,
            command=base.messenger.send,
            extraArgs= ["Credits-Back"],
            pressEffect=False,
            rolloverSound=None,
            clickSound=None)
        self.btnBack.setTransparency(1)
        self.btnBack.reparentTo(self.frameMain)

        self.hide()

    defshow(self):
        self.frameMain.show()
        self.creditsScroll.loop()

    defhide(self):
        self.frameMain.hide()
        self.creditsScroll.finish()
```

Setup File code

```
fromsetuptoolsimportsetup

exclude = [
    # build stuff
    'build/**',
    'build',
    'dist/**',
    'dist',
    '**/*.py',
    'setup.py',
    'requirements.txt']

setup(
    name='K.O game',
    author="University of central punjab",
    author_email="imran.hussain@ucp.edu.pk",
    options= {
        'build_apps': {
            'include_patterns': ['**/*'],
            'exclude_patterns': exclude,
            'gui_apps': {
                'K.O Game': 'main.py',
            },
            'plugins': [
                'pandagl',
                'p3openal_audio'
            ]
        }
    }
)
```

Assets Images

Characters

Arenas Images