import viz

import vizshape

import vizfx

import vizact

import vizinfo

import random

viz.go()

##### Set background and lighting

viz.clearcolor(viz.SKYBLUE) # Sky blue background

light = viz.addLight()

light.position(0, 10, 0) # Light source above the field

##### Add a plane for the football field

grass = viz.addTexture('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR1/grass.jpg')

field = vizshape.addPlane(size=(150, 100)) # Field size: 100X60 units

field.setPosition(0, 0, 0) # Center the field

field.setEuler(0, 0, 0) # Rotate to lay flat

field.texture(grass) # Apply the ice texture

##### Add field lines (boundaries and center line)

# Top Boundary

boundary\_top = vizshape.addBox(size=(100/4, 0.01, 0.1))

boundary\_top.setPosition(0, 0.01, 60/8) # At the top edge

boundary\_top.color(viz.WHITE)

# Bottom Boundary

boundary\_bottom = vizshape.addBox(size=(100/4, 0.01, 0.1))

boundary\_bottom.setPosition(0, 0.01, -60/8) # At the bottom edge

boundary\_bottom.color(viz.WHITE)

# Left Boundary

boundary\_left = vizshape.addBox(size=(0.1, 0.01, 60/4))

boundary\_left.setPosition(-100/8, 0.01, 0) # Left edge

boundary\_left.color(viz.WHITE)

# Right Boundary

boundary\_right = vizshape.addBox(size=(0.1, 0.01, 60/4))

boundary\_right.setPosition(100/8, 0.01, 0) # Right edge

boundary\_right.color(viz.WHITE)

##### Add goalposts

goal\_left = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

goal\_right = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

goal\_top = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

## Place the left goalpost

goal\_left.setPosition((-100/8)-(1/3), 0.01, 0) # Position at the left goal

goal\_left.setEuler(-90, 0, 0) # Rotate to face the field

bbox = goal\_left.getBoundingBox(viz.ABS\_GLOBAL) # Get bounding box in global coordinates

width = bbox.width

height = bbox.height

depth = bbox.depth

# Target size

target\_width = 8\*1.75/4

target\_height = 8\*1.75/(2\*3)

target\_depth = 8\*1.75/(3)

# Calculate scale factors

scale\_x = target\_width / width

scale\_y = target\_height / height

scale\_z = target\_depth / depth

goal\_left.setScale(scale\_x, scale\_y, scale\_z)

bbox = goal\_left.getBoundingBox(viz.ABS\_GLOBAL)

print("Final dimensions (width, height, depth):",

bbox.width, bbox.height, bbox.depth)

# Place the right goalpost

goal\_right.setPosition((100/8)+(1/3), 0.01, 0) # Position at the right goal

goal\_right.setEuler(90, 0, 0) # Rotate to face the field

goal\_right.setScale(scale\_x, scale\_y, scale\_z)

# Place the top goalpost

goal\_top.setPosition(0, 0.01, (60/8)+(1/3)) # Position at the top goal

goal\_top.setEuler(0, 0, 0) # Rotate to face the field

goal\_top.setScale(scale\_x, scale\_y, scale\_z)

# Add a hockey ball at the center

ball = viz.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/puck/hockey\_ball.fbx') # Load the hockey ball OBJ file

ball.setPosition(0, 0.01, 0) # Center the ball on the field

ball.setEuler(0, 90, 0) # Adjust rotation if needed

ball.setScale(0.3, 0.3, 0.3) # Scale the ball to match the field size

# Add a hockey stick near the ball

hockey\_stick = viz.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_stick/hockey\_stick.fbx') # Load the hockey stick OBJ file

hockey\_stick.setPosition(0,0.01, 0) # Place the stick near the ball

hockey\_stick.setEuler(0, 0, 0) # Adjust rotation if needed

hockey\_stick.setScale(0.1, 0.1, 0.1) # Scale the stick to match the ball

# Add a white circle on the ball

ball\_circle = vizshape.addCircle(radius=0.2) # Adjust the radius as needed

ball\_circle.setPosition(0, 0.01, 0) # Slightly above the ball's center

ball\_circle.setEuler(0, 90, 0) # Keep it flat

ball\_circle.color(viz.WHITE) # Set the color to white

##### Adjust camera position to view the entire field

viz.MainView.setPosition(0, 20, -20) # Camera above and behind the field

viz.MainView.setEuler(0, 40, 0) # Tilt the camera to look down

viz.MainView.collision(False)

# Variables to track game state

# Variables to track game state

field\_width = (100/8)+(4/3) # Half the field size (assumes 100x60 yards field)

field\_depth = (60/8)+(4/3)

score = 0

current\_task = 0 # 0: left goal, 1: top goal, 2: right goal

global goal\_positions

goal\_positions = [

(-100/8, 0.2, 0), # Left goal

(0, 0.2, 60/8), # Top goal

(100/8, 0.2, 0) # Right goal

]

# Global variables

task\_order = [] # List to hold the randomized task order

def setupGame():

global task\_order

# Create a list of task indices [0, 1, 2] representing Left, Top, Right

task\_order = [0, 1, 2]

random.shuffle(task\_order) # Shuffle the list to randomize the order

print(f"Randomized task order: {task\_order}")

# to randomise the tasks

setupGame()

ball\_movement\_enabled = False

# Global variable to track the current instruction message

current\_instruction = None

def showInstructionMessage(message, color, duration):

"""Display an on-screen instruction message, replacing any existing one."""

global current\_instruction

# Remove the previous instruction if it exists

if current\_instruction:

current\_instruction.remove()

current\_instruction = None

screen\_width, screen\_height = viz.MainWindow.getSize() # Get the screen dimensions

# Adjust positions for the top-left corner

# Using normalized screen coordinates: x=[0,1] and y=[0,1]

msg\_pos = [0.03, 0.90, 0] # Slightly below celebration

# Display the new instruction

current\_instruction = viz.addText(message, parent=viz.SCREEN, pos=msg\_pos, scale=[0.3, 0.4, 0.4])

current\_instruction.color(color)

# Automatically remove the instruction after the specified duration

vizact.ontimer2(duration, 0, lambda: removeCurrentInstruction())

def removeCurrentInstruction():

"""Remove the current instruction message."""

global current\_instruction

if current\_instruction:

current\_instruction.remove()

current\_instruction = None

# At the beginning of the game (Level 1 start)

showInstructionMessage("Level 1 Starting!", color=viz.BLACK, duration=3)

def showScoreboardAndCelebration(task\_score, total\_score, is\_goal):

"""Show the scoreboard and celebration animation for a completed task in the top-right corner."""

screen\_width, screen\_height = viz.MainWindow.getSize() # Get the screen dimensions

# Adjust positions for the top-right corner

# Using normalized screen coordinates: x=[0,1] and y=[0,1]

celebration\_pos = [0.75, 0.95, 0] # Top-right for celebration

scoreboard\_pos = [0.75, 0.90, 0] # Slightly below celebration

# Display 'Task complete!'

celebration\_text = "Task complete!"

celebration = viz.addText(celebration\_text, parent=viz.SCREEN, pos=celebration\_pos, scale=[0.4, 0.5, 0.5])

celebration.color(viz.BLACK)

# Display task score and total score

scoreboard\_text = f"Task Score: {task\_score}\nTotal Score: {total\_score}"

scoreboard = viz.addText(scoreboard\_text, parent=viz.SCREEN, pos=scoreboard\_pos, scale=[0.3, 0.4, 0.4])

scoreboard.color(viz.BLACK)

# Automatically remove the texts after a delay

vizact.ontimer2(3, 0, celebration.remove) # Remove celebration text

vizact.ontimer2(3, 0, scoreboard.remove) # Remove scoreboard text

# Load an arrow model

arrow\_model = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR1/arrow.fbx')

# Scale and position the arrow

arrow\_model.setScale(0.02, 0.02, 0.02) # Adjust the size

arrow\_model.setPosition(0, 0, 0) # Place it above the field

arrow\_model.visible(viz.OFF)

# New variables to track ball stay time inside the goalpost

ball\_inside\_goal = False

time\_inside\_goal = 0 # Time the ball has stayed inside the goalpost

goal\_pause\_duration = 3 # Duration the ball pauses after scoring a goal

stay\_required\_time = 1 # Time the ball needs to stay for full points

last\_corner\_state = False # Tracks whether the ball was in a corner inside the goal

##### Check if the ball enters the goal or goes out of bounds

def checkGoal():

global score, current\_task, field\_width, field\_depth, break\_in\_progress, game\_complete, ball\_inside\_goal, time\_inside\_goal, last\_corner\_state

if game\_complete or break\_in\_progress:

return # Skip checks if the game is complete or during the break

ball\_pos = ball.getPosition()

# Check if all tasks are completed

if current\_task >= len(goal\_positions):

if field\_width == field\_width\_level2:

print("Level 2 complete! Congratulations!")

vizact.ontimer2(0.05, 0, lambda: viz.quit()) # Exit the program after a short delay

else:

print("Level 1 complete! Congratulations!")

switchToLevel2()

return

# Ensure the current task index is within bounds

if current\_task < len(goal\_positions):

goalpost = goal\_positions[current\_task]

# Get the current task based on the randomized order

current\_task\_index = task\_order[current\_task]

arrow\_model.visible(viz.ON)

# Define goalpost based on the shuffled current task index

if current\_task\_index == 0:

arrow\_model.setEuler(0, 90, 90)

arrow\_model.setPosition(-2, 0, 0) # Place it above the field

goalpost = goal\_left if field\_width == (100/8)+(4/3) else goal\_left\_level2

elif current\_task\_index == 1:

arrow\_model.setEuler(0, 90, 0)

arrow\_model.setPosition(0, 0, 2) # Place it above the field

goalpost = goal\_top if field\_depth == (60/8)+(4/3) else goal\_top\_level2

else:

arrow\_model.setEuler(0, 90, -90)

arrow\_model.setPosition(2, 0, 0) # Place it above the field

goalpost = goal\_right if field\_width == (100/8)+(4/3) else goal\_right\_level2

# Obtain the bounding box of the goalpost in world coordinates

bounding\_box = goalpost.getBoundingBox(viz.ABS\_GLOBAL)

goal\_min = [bounding\_box.xmin, bounding\_box.ymin, bounding\_box.zmin]

goal\_max = [bounding\_box.xmax, bounding\_box.ymax, bounding\_box.zmax]

# Calculate the 20% area of the goal (corners)

corner\_width = ((field\_width-(4/3))\*2-7)/4

corner\_depth = ((field\_depth-(4/3))\*2-7)/4

# Determine the goalpost type and check ball position accordingly

if current\_task\_index == 0: # Left goalpost

is\_in\_left\_corner = False

is\_in\_right\_corner = False # No right corner in left goal

is\_in\_bottom\_corner = (

(goal\_min[2] >= ball\_pos[2] >= goal\_min[2]-corner\_depth and

goal\_min[0] <= ball\_pos[0] <= goal\_max[0])

)

is\_in\_top\_corner = (

(goal\_max[2] <= ball\_pos[2] <= goal\_max[2]+corner\_depth and

goal\_min[0] <= ball\_pos[0] <= goal\_max[0])

)

elif current\_task\_index == 1: # Top goalpost

is\_in\_left\_corner = (

(goal\_min[0] >= ball\_pos[0] >= goal\_min[0]-corner\_depth and

goal\_min[2] <= ball\_pos[2] <= goal\_max[2])

)

is\_in\_right\_corner = (

(goal\_max[0] <= ball\_pos[0] <= goal\_max[0]+corner\_depth and

goal\_min[2] <= ball\_pos[2] <= goal\_max[2])

)

is\_in\_top\_corner = False # No top corner in the top goal

is\_in\_bottom\_corner = False # No bottom corner in the top goal

elif current\_task\_index == 2: # Right goalpost

is\_in\_left\_corner = False # No left corner in the right goal

is\_in\_right\_corner = False

is\_in\_bottom\_corner = (

(goal\_min[2] >= ball\_pos[2] >= goal\_min[2]-corner\_depth and

goal\_min[0] <= ball\_pos[0] <= goal\_max[0])

)

is\_in\_top\_corner = (

(goal\_max[2] <= ball\_pos[2] <= goal\_max[2]+corner\_depth and

goal\_min[0] <= ball\_pos[0] <= goal\_max[0])

)

# Check if the ball is within the goalpost bounding box

is\_inside\_goal = (

goal\_min[0] <= ball\_pos[0] <= goal\_max[0] and

goal\_min[1] <= ball\_pos[1] <= goal\_max[1] and

goal\_min[2] <= ball\_pos[2] <= goal\_max[2]

)

if is\_inside\_goal or is\_in\_left\_corner or is\_in\_right\_corner or is\_in\_top\_corner or is\_in\_bottom\_corner:

if not ball\_inside\_goal:

ball\_inside\_goal = True

time\_inside\_goal = 0 # Reset stay timer

print("Ball entered the goalpost area. Timer started.")

# Increment the stay timer if ball stays inside

time\_inside\_goal += viz.elapsed() # Increment timer by elapsed time

# Update corner state

last\_corner\_state = (

is\_in\_left\_corner or is\_in\_right\_corner or

is\_in\_top\_corner or is\_in\_bottom\_corner

)

if time\_inside\_goal >= stay\_required\_time:

if is\_in\_left\_corner or is\_in\_right\_corner or is\_in\_top\_corner or is\_in\_bottom\_corner:

# Award 60 points if the ball is in the corner

task\_score = 80

score += task\_score

print(f"Ball entered the corner! You earned {task\_score} points. Total score: {score}.")

showScoreboardAndCelebration(task\_score, score, True)

current\_task += 1

pauseBallInGoal() # Pause the ball for the goal pause duration

else:

task\_score = 100

score += task\_score

print(f"Goal! You earned {task\_score} points. Total score: {score}.")

showScoreboardAndCelebration(task\_score, score, True)

current\_task += 1

pauseBallInGoal() # Pause the ball for the goal pause duration

if current\_task==3 and field\_width == (100/8)+(4/3):

showInstructionMessage("Level 1 Complete! Congratulations!", color=viz.BLACK, duration=3)

elif current\_task==3 and field\_width != (100/8)+(4/3):

showInstructionMessage("Level 2 Complete! Congratulations!", color=viz.BLACK, duration=3)

return

else:

# Reset the state for the next check

ball\_inside\_goal = False

if time\_inside\_goal >0 and time\_inside\_goal < stay\_required\_time:

# Ball left prematurely before reaching the required stay time

if last\_corner\_state:

task\_score = 60

else:

task\_score = 80

score += task\_score

print(f"Ball left the goal early. You earned {task\_score} points. Total score: {score}.")

showScoreboardAndCelebration(task\_score, score, False)

startBreak() # Reset ball to the center

current\_task += 1

time\_inside\_goal = 0

last\_corner\_state = False

# Check if the ball goes out of bounds

if abs(ball\_pos[0]) > field\_width-0.5 or abs(ball\_pos[2]) > field\_depth-0.5:

task\_score = 60

score += task\_score

print(f"Ball went out of bounds! Your score {score}. Task {current\_task + 1} failed.")

showScoreboardAndCelebration(task\_score, score, False) # Show scoreboard without celebration

current\_task += 1

if current\_task==3 and field\_width == (100/8)+(4/3):

showInstructionMessage("Level 1 Complete! Congratulations!", color=viz.BLACK, duration=3)

elif current\_task==3 and field\_width != (100/8)+(4/3):

showInstructionMessage("Level 2 Complete! Congratulations!", color=viz.BLACK, duration=3)

startBreak() # Start the 5-second break

return

def resetGoalTimer():

"""Reset the goal timer."""

global goal\_timer, goal\_timer\_active

goal\_timer = 0

goal\_timer\_active = False

def startBreak():

"""Initiate a 5-second break and reset the ball."""

global break\_in\_progress

break\_in\_progress = True

resetBallPosition()

print("Taking a 5-second break...")

vizact.ontimer2(5, 0, endBreak) # Resume after 5 seconds

def endBreak():

"""End the break and allow the game to continue."""

global break\_in\_progress

break\_in\_progress = False

if current\_task < len(goal\_positions): # Only print if there are valid tasks remaining

print(f"Task {current\_task + 1} starting now!")

def pauseBallInGoal():

"""Pause the ball in the goal for a specified duration and then reset it."""

global ball\_movement\_enabled

global break\_in\_progress

break\_in\_progress = True

ball\_movement\_enabled = False # Disable ball movement

print("Ball paused in the goal for celebration.")

# Wait for the pause duration, then reset

vizact.ontimer2(goal\_pause\_duration, 0, resetBallPosition)

print("Taking a 5-second break...")

vizact.ontimer2(5, 0, endBreak) # Resume after 5 seconds

# Reset ball position to center

def resetBallPosition():

"""Reset ball position to the center of the field and disable movement."""

global ball\_movement\_enabled, ball\_inside\_goal

ball\_inside\_goal = False

ball.setPosition(0, 0.1, 0) # Center the ball

hockey\_stick.setPosition(0, 0.1, 0) # Place stick near the ball

ball\_movement\_enabled = False # Ensure movement remains disabled

print("Ball reset to center position.")

##### Initially disable ball movement

def onKeyDown(key):

global ball\_movement\_enabled

if key == viz.KEY\_RETURN or key == " ": # Enter or Space key

ball\_movement\_enabled = not ball\_movement\_enabled # Toggle movement

if ball\_movement\_enabled:

print("Ball movement enabled.")

else:

print("Ball movement paused.")

### Update ball position based on mouse input

def updateBallPosition():

viz.callback(viz.KEYDOWN\_EVENT, onKeyDown)

global current\_task

# Prevent ball movement during break

if break\_in\_progress or not ball\_movement\_enabled:

return # Skip updating ball position

mouse\_x = viz.mouse.getPosition()[0] \* 2 - 1 # Scale and center

mouse\_z = viz.mouse.getPosition()[1] \* 2 - 1 # Scale and center

# Map mouse position to field size

ball\_x = mouse\_x \* field\_width

ball\_z = mouse\_z \* field\_depth

# Update ball position

ball.setPosition(ball\_x, 0.1, ball\_z)

hockey\_stick.setPosition(ball\_x, 0.1, ball\_z) # Place stick near the ball

if task\_order[current\_task]==0:

hockey\_stick.setEuler(-90, 0, 0) # Adjust rotation if needed

# Update stick position near the ball

hockey\_stick.setPosition(ball\_x+0.5, 0.1, ball\_z) # Place stick near the ball

if task\_order[current\_task]==1:

hockey\_stick.setEuler(0, 0, 0) # Adjust rotation if needed

# Update stick position near the ball

hockey\_stick.setPosition(ball\_x , 0.1, ball\_z) # Place stick near the ball

if task\_order[current\_task]==2:

hockey\_stick.setEuler(-270, 0, 0) # Adjust rotation if needed

# Update stick position near the ball

hockey\_stick.setPosition(ball\_x-0.5, 0.1, ball\_z) # Place stick near the ball

# Global variable to track if a break is in progress

break\_in\_progress = False

# Add this global variable

game\_complete = False

# Level 2 configurations

field\_width\_level2 = (100/6)+(4/3) # Half the field size for Level 2

field\_depth\_level2 = (60/6)+(4/3)

goal\_positions\_level2 = [

(-100/6, 0.2, 0), # Left goal for Level 2

(0, 0.2, 100/6), # Top goal for Level 2

(100/6, 0.2, 0) # Right goal for Level 2

]

# Goalpost and boundary objects for Level 2

goal\_left\_level2 = None

goal\_right\_level2 = None

goal\_top\_level2 = None

boundary\_objects\_level2 = []

def setupLevel2():

global goal\_left\_level2, goal\_right\_level2, goal\_top\_level2, boundary\_objects\_level2

# Create larger goalposts for Level 2

goal\_left\_level2 = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

goal\_right\_level2 = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

goal\_top\_level2 = vizfx.addChild('C:/IITGN 2nd year/Muscle Synergy Project/Vizard/VR2/hockey\_gate/hockey\_gate.fbx')

# Position and scale Level 2 goalposts ((115 X 74)/3 yards)

goal\_left\_level2.setPosition((-100/6)-(1/3), 0.01, 0)

goal\_left\_level2.setScale(scale\_x, scale\_y, scale\_z)

goal\_left\_level2.setEuler(-90, 0, 0)

goal\_right\_level2.setPosition((100/6)+(1/3), 0.01, 0)

goal\_right\_level2.setScale(scale\_x, scale\_y, scale\_z)

goal\_right\_level2.setEuler(90, 0, 0)

goal\_top\_level2.setPosition(0, 0.01, (60/6)+(1/3))

goal\_top\_level2.setScale(scale\_x, scale\_y, scale\_z)

goal\_top\_level2.setEuler(0, 0, 0)

# Create larger boundaries for Level 2

boundary\_objects\_level2 = [

vizshape.addBox(size=(100/3, 0.01, 0.1)), # Top boundary

vizshape.addBox(size=(100/3, 0.01, 0.1)), # Bottom boundary

vizshape.addBox(size=(0.1, 0.01, 60/3)), # Left boundary

vizshape.addBox(size=(0.1, 0.01, 60/3)) # Right boundary

]

boundary\_objects\_level2[0].setPosition(0, 0.01, 60/6) # Top boundary

boundary\_objects\_level2[1].setPosition(0, 0.01, -60/6) # Bottom boundary

boundary\_objects\_level2[2].setPosition(-100/6, 0.01, 0) # Left boundary

boundary\_objects\_level2[3].setPosition(100/6, 0.01, 0) # Right boundary

for boundary in boundary\_objects\_level2:

boundary.color(viz.WHITE)

def switchToLevel2():

global field\_width, field\_depth, goal\_positions, current\_task

# Hide Level 1 objects

goal\_left.visible(False)

goal\_right.visible(False)

goal\_top.visible(False)

boundary\_top.visible(False)

boundary\_bottom.visible(False)

boundary\_left.visible(False)

boundary\_right.visible(False)

# Setup Level 2

setupLevel2()

# Update game state for Level 2

field\_width = field\_width\_level2

field\_depth = field\_depth\_level2

goal\_positions[:] = goal\_positions\_level2

current\_task = 0

# Adjust camera viewpoint for Level 2

viz.MainView.setPosition(0, 30, -30) # Higher and farther camera position

viz.MainView.setEuler(0, 40, 0) # Tilt camera to look down

# Reset ball position and disable movement

resetBallPosition()

ball\_movement\_enabled = False

print("Welcome to Level 2! The field and goalposts have expanded!")

showInstructionMessage("Welcome to Level 2!", color=viz.BLACK, duration=3)

# to randomise the tasks

setupGame()

# Register continuous callbacks

vizact.ontimer(0, updateBallPosition)

vizact.ontimer(0.1, checkGoal)