**Software Architecture & Design (SAD) Document v.1.X**

**Project Name: Breath of the Coyote Date: May 11th, 2020**

**Project Manager: Cameron Maclean**

**Assistant Project Manager: Stephen Anderson**

**Software Engineer #1: Jugen Fornoles - Combat**

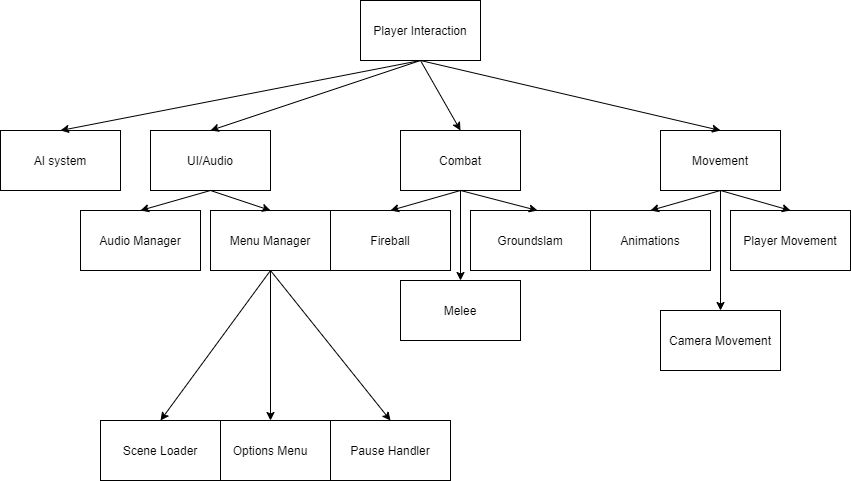
**Software Engineer #2: Valentino Fernandes - UI/Audio**

**Software Engineer #3: Christian Morales - Movement**

**Software Engineer #4: Brandon Edmonds - AI system**

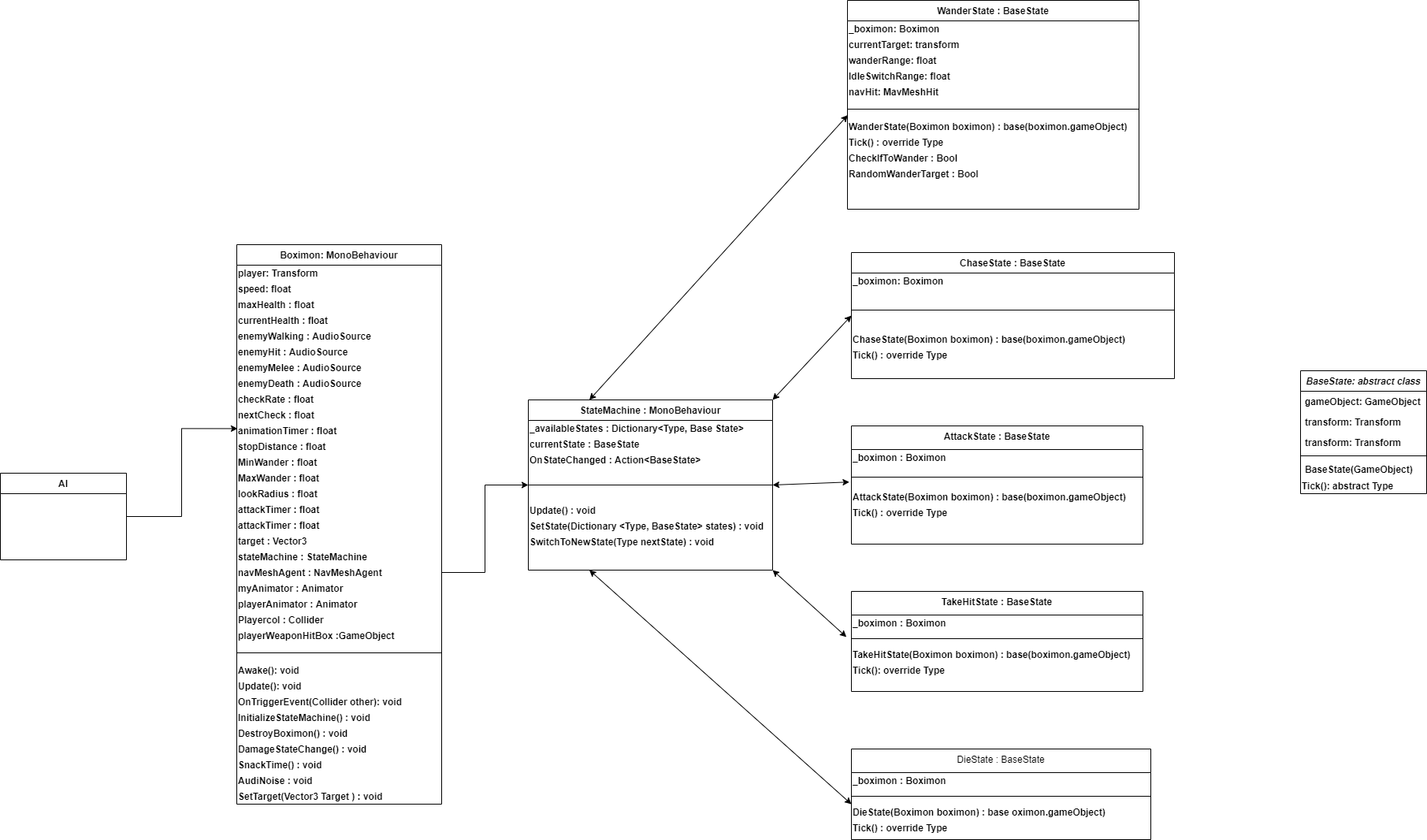
**1. Purpose – The purpose of this document is to preserve and maintain the Breath of the Coyote mobile game. It will be done by depicting a visual representation via diagrams and written pseudo-code. The diagrams will be a High-Level Architecture diagram representing the player interaction with the key systems that drive the game and a Low-Level Architecture diagram of these key systems. The pseudo-code will be based on critical functions used in the software.**

**2. High-Level Design – Created in Hierarchical decomposition**

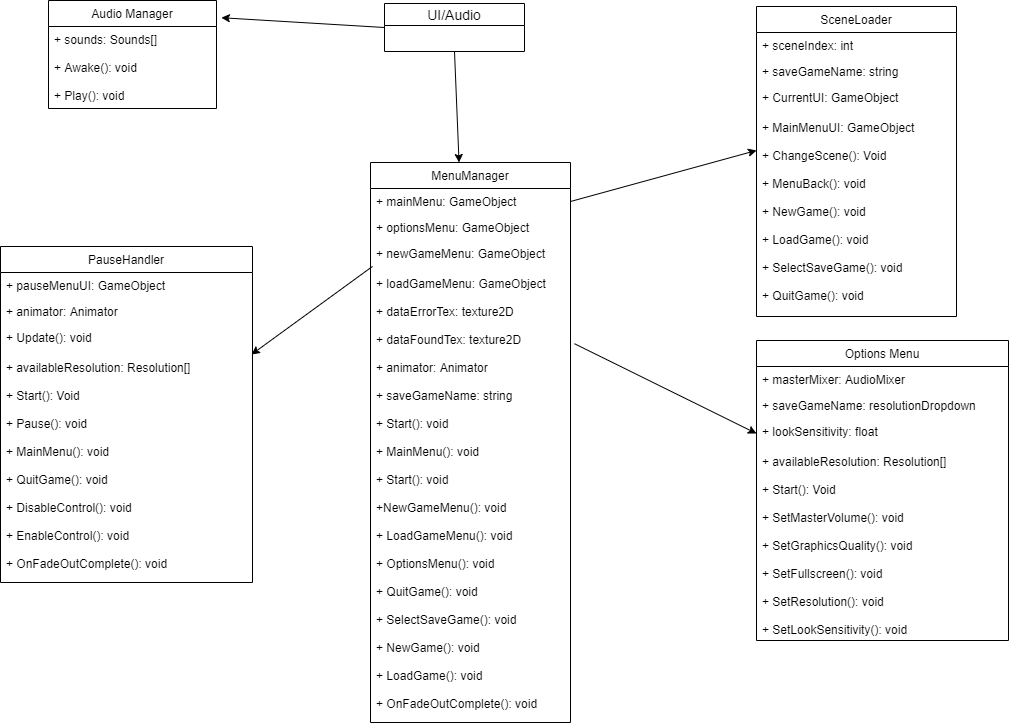
****

**3. Low-Level Design –**

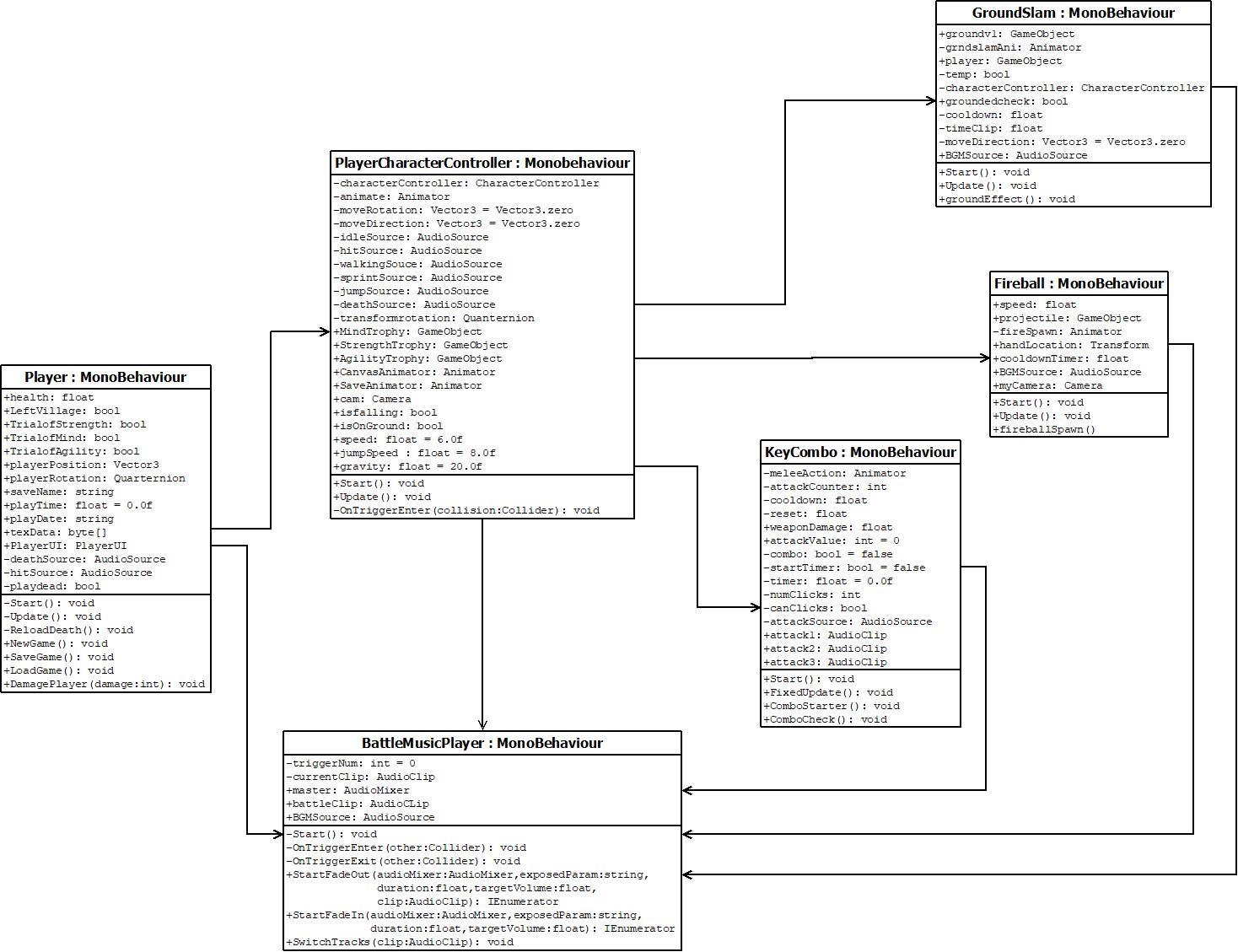
**AI system -**

****

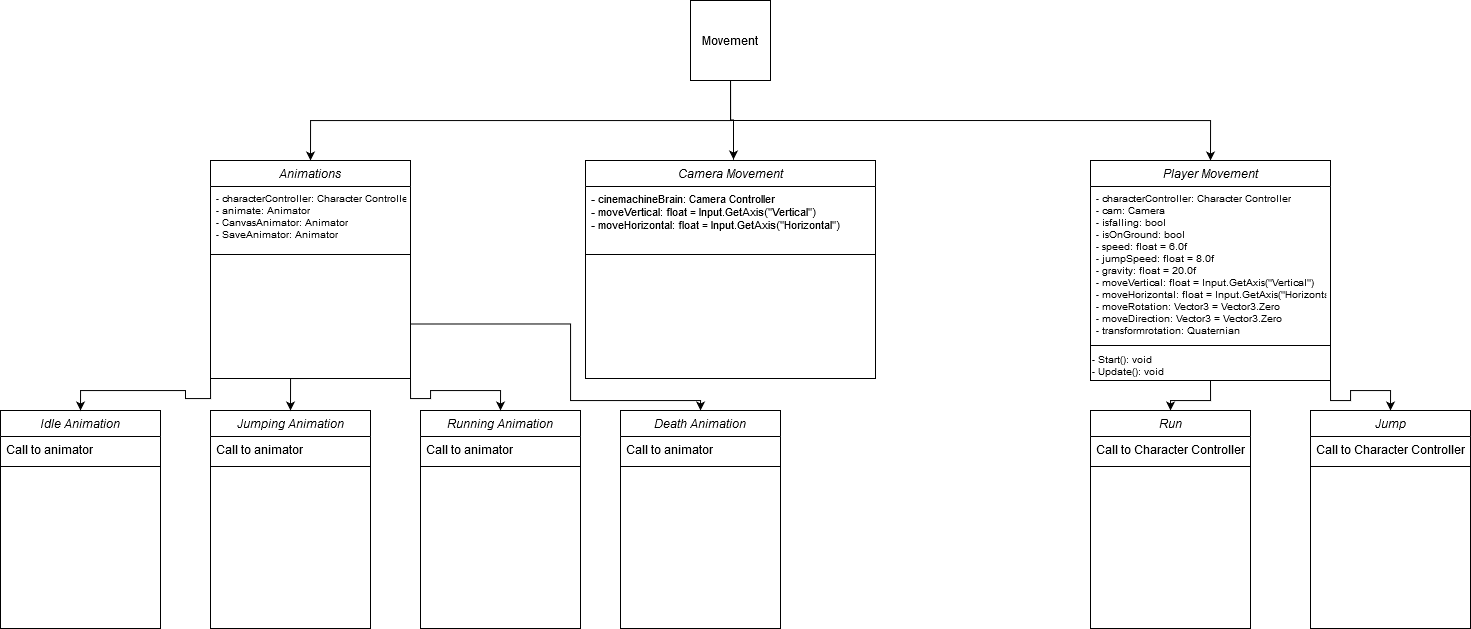
**UI/Audio -**



**Combat -**

****

**Movement -**

****

**4. Pseudo-Code Algorithms of Critical Functions in Class Diagram –**

**AI system -**

Base State Class

Override Type Tick(){

Abstract function that State Machine Script calls to run actions within a specified state.  
 }

StateMachine Class

void Update(){

If there is no current state{

Currentstate variable is equal to first state in enemy characters library

}

The next state variable is equal to the current states ’Tick’ function

If the nextstate variable is not null and is not equal to old state type{  
 SwitchToNewState(next state variable)

}

}

SwitchToNewState(Type nextState){

Set current state to a state that matches the Type of states that were initialized in beginning of run time

Invoke tick method for the new current state.

}

Boximon Class

InitializeStateMachine(){  
 When called takes all states for Boximon and inserts them to the state machine dictionary

}

void Awake(){

Calls InitiaizeStateMachin function and sets boximon attributes

}

void Update(){

Keeps track of player, movement speed, and countdown for animationTimer

}

private void OnTriggerEnter(){

If the animation timer is less than 0 and collided object is the players weapon hitbox{

Turn off all animation parameters

Call DamageStateChangeFunction{

}

}

private void DamageStateChange(){

Creates temporary dictionary of States that only include ‘TakeHit’ and DieState

If current health is less than or equal to 0{  
 Use SwitchToNewState function to TakeHitState

}

else{

Use SwitchToNewState function to DieState

}

}

**UI/Audio -**

Audio-Manager Pseudo Code:

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Using typical Unity library + Unity Audio

public class AudioManager : MonoBehaviour

{

Make an array of sounds

// Start is called before the first frame update

On wake up

{

For every sound in the list

{

Make it a game object

Clip the audio

Give it volume and pitch

Loop the track

}

}

Make a Play() function that requires a string

{

The string is the search condition for the array of sounds

if(s == null)

{

If no sound that matches the string is called tell the player that they messed up

}

Otherwise, play the song

}

}

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

**Combat -**

PlayerCharacterController Class

void Start(){

Get the gameobject of the character

Get the animator of the character

Setting all the three trophies to default(false); the player starts with no trophies

Get all the corresponding AudioSources of the players actions within the game

}

void Update(){

//this is for character movement which is an essential part of the game and combat

If character is moving{

Set walking to true

If character is sprinting{

Multiply speed by 1.5

Play audio and animation for running

}

Else{

Walk with normal speed

Play audio and animation for walking

}

}

Else{ //character is not moving

Set the state of the character to be Idle or walking to false

Play audio and animation for idle

}

}

private void OnTriggerEnter(Collider collision){

check for each trophy:

check if the player has Agility Trophy

check if the player has Mind Trophy

check if the player has Strength Trophy

}

GroundSlam class

void Update(){

set the animation to false

increment cooldown using time

if GroundSlam key is pressed and the cooldown is morethan 2 units{

set the animation to true

}

Else{

Do nothing

}

}

void groundEffect(){

special effects are instantiated

destroy the special effects

set the animation to false

set the cooldown to zero

}

Fireball Class

void Update(){

set fireballspawn to false //not fireball visible

increment the cooldown with time

If enough cooldown and fireball key is pressed{

Set cooldown to zero and spawn a fireball

}

Else{

Do nothing

}

}

void fireballSpawn(){

instantiate the fireball bullet

Play the AudioSource

Set distance for the aim for the fireball

Then make the bullet moveforward

}

KeyCombo Class

void FixedUpdate(){

once the left button of the mouse is clicked once trigger ComboStarter()

}

public void ComboStarter(){

set the animator parameter isAttacting to true

if canClick is available{ //animation is not yet rendered

numClick is incremented

}

If numClick is equals to 1{

Set the value of AttackValue to 1 // this is the start of the combos

Play attack1 animation and audio

}

}

public void ComboCheck(){ //this is being called after the animation is played via animation event

set canClick to false //stopping the numClick from incrementing

if numClick is 1 and attack1 animation is being rendered{

End combo here

}

Else If numClick is greater than or equals to 2 and attack1 animation is being rendered{

Set attackvalue to 2

Play attack2 animation and audio

}

Else if numClick is 2 and attack2 animation is being rendered{

End combo here

}

Else if numClick is greater than or equals to 3 and attack2 animation is being rendered{

Set attackvalue to 3

Play attack3 animation and audio

}

Else if attack3 animation is being rendered{

End combo here

Set canClick to true //to be able to start another combo

Reset numClick to zero

}

}

**Movement -**

PlayerCharacterController Class

void Start(){

Get the gameobject of the character

Get the animator of the character

Setting all the three trophies to default(false); the player starts with no trophies

Get all the corresponding AudioSources of the players actions within the game

}

public void Bool()

{

Set the jumping bool to true for the touchscreen jump button to know when to activate onclick.

}

void Update(){

//this is for character movement which is an essential part of the game and combat

If character is moving{

Set walking to true

If character is sprinting{

Multiply speed by 1.5

Play audio and animation for running

}

Else{

Walk with normal speed

Play audio and animation for walking

}

}

Else{ //character is not moving

Set the state of the character to be Idle or walking to false

Play audio and animation for idle

}

if the character is currently on the ground{

The bool used for the animator to know if we’re jumping is set to false

The audio source for jumping is set to not play

The vector 3 for controlling rotation is set to the input axes

The move direction for controlling movement is set to the input axes

if the jump button has been pressed{

Stop the running animation

Set the jumping bool to false to let the mobile buttons you are already jumping

Give the player speed in the air relative to the movement speed they had on ground

Start the jumping animation

Play the jumping sound effect from the audio source

Stop the walking sound effect

}

}

Else{ //The player is not grounded but not jumping

The vector 3 for controlling rotation is set to the input axes

}

}

private void OnTriggerEnter(Collider collision){

check for each trophy:

check if the player has Agility Trophy

check if the player has Mind Trophy

check if the player has Strength Trophy

}