Notes:

April 18:

-When working in 2D, must remove the “Box Collider” component and replace it with the “Box Collider 2D” component for it to function properly

-Must add “Rigidbody 2D” to an object to give it “gravity”

-Organize your assets using folders in the assets folder

-For names of objects and scripts, use multiple words combined into one, with individual words separated by Capital letters: “CopterJump,” etc.

-Awake activates the moment the game starts, Start activates once the object the script is attached to becomes enabled

-GetKey: the function will activate for as long as the user holds the key down

-GetKeyDown: the function will only activate upon the user first pressing the key (the key must be released first to do it again). Use this more.

-Use private variables to make debugging later on easier

-Any public variables in a script can be manually changed in the inspector of an attached object

-To make an object jump, you have to create a new vector that describes how the object will move (in this case, just up), create a public rigidbody(2D) variable, drag the ridigbody2D component of the object in question to the script component in the Inspector, and then do .addForce

-For moving left or right, use transform.position (so that the movement is smooth, not influenced by physics too much)

-use a linecast when I now what distance I'm supposed to check from my character (returns true if there's anything intersecting the collider between start and end

April 19:

controlling audio: <http://www.kaappine.fi/tutorials/using-microphone-input-in-unity3d/>

original code for jumping with rigidbody:

/\*

//jumping

if ( (Input.GetKeyDown("space") || Input.GetKeyDown("up")) && grounded)

{

Vector3 newPush = new Vector3(0, jumpForce, 0);

rb2d.AddForce(newPush);

noMove = true; //we disable the abiltiy to move left and right while jumping

beginCountDown = true; //as soon as the character is launched into the air, we start a countdown timer

}

//landing

//we begin a timer until when we can move in the air...

if (beginCountDown)

{

airTime -= Time.deltaTime;

}

//we make it so that the character can move in the air, and slow its descent

if(airTime <= 0)

{

noMove = false;

beginCountDown = false;

Vector3 newPushBack = new Vector3(0, gravityForce, 0); //NOTE: when I fall, my default gravity is Vector3(0,-9.81,0)

rb2d.AddForce(newPushBack); //the player's gravity is reduced here

}

\*/

-use time.time (time elapsed since the start of the program) instead of time.deltaTime (time elapsed since last frame)

-in the inspector, go to constraints > freeze rotation to prevent the object in question from rotating

(I should do a “why I'm making games now” doc to motivate/remind myself)

April 21:

-If you have multiple layers of child objects, and the 2nd child object is attached to the first via a hinge joint, but the 1st child object is not attached to the parent via a hinge joint, the 2nd child will behave awkwardly

-Increase linear drag to make a rigidbody object attached to a hinge joint to sway left to right less

-Check “auto configure connection” to make sure that the rotor blade remains connected to the rotor mast with the hinge joint

-And don’t attach a Connected Rigidbody (otherwise it won’t move)

/\* alternate code to try and proportionally push down on the player

//((tempTime+airTime) - tempTimeForCompensation) = time remaining since the jump began

//(((tempTime+airTime) - tempTimeForCompensation)/airTime) = the percentage of time the player did not spend ascending, and what to multiply jumpforce by

Vector3 newPush = new Vector3(0, -jumpForce\*((((tempTime + airTime) - tempTimeForCompensation) / airTime))\*0.01f, 0);

rb2d.AddForce(newPush);

print(-jumpForce \* ((((tempTime + airTime) - tempTimeForCompensation) / airTime))\*0.5f);

\*/

April 24:

-In order to add text to the screen, must use a canvas (or if I directly put in text, it is automatically made the child of the canvas)

-Strongly reccomended that I then set the Canvas’ render mode to Screen Space – Camera so that it appears much smaller in the scene view, and is tied to the current camera in use

-IMP: in the Canvas object’s inspector, go to Canvas Scaler (Script), change constant pixel size (which keeps UI elements at their initial size; does not resize them when screen resolution changes) to scale with screen size

-Go to edit – project settings – quality to set quality settings

-Go to edit – project settings – player to set the screen resolution

-In player settings, go to supported aspect ratios to restrict what kind of aspect ratios the player can use when playing the game (IMP, as wildly different aspect ratios have a tendency to reveal more on-screen objects, and hide others, impacting the game)

April 26:

-Must add rigidbody to both objects I am trying to collide

-Lerp creates a movement in which the object goes from fast to slow, not a consistent speed

-Use Vector3.MoveTowards instead

May 2:

-The Quaternion equivalent for Vector3.Movetowards is Quaternion.RotateTowards

-When using the hingejoint motor, you have to uncheck/disable the “use motor” option to make it ragdoll again: hinge.useMotor = false;

May 9:

-Use invokerepeating to repeat an action every few seconds, after a starting time:

InvokeRepeating("LaunchProjectile", 2.0f, 0.3f);

(the first part is a string of the method’s name, the second part is how many seconds until it starts, the third is long the interval is until the action is performed

-BUT IT ONLY WORKS IF PUT INTO THE START

**(LOTS OF COMMENTS SLOW DOWN UNITY’S PREVIEW MODE!)**

-Old code for making the thrown blades move back to the rotor mast:

/\*

if(transform.position == GameObject.Find("CopterRotorMast").transform.position && returnHome)

{

if (identityValue == 1)

{

GameObject.Find("LooseCopterBlade1").GetComponent<looseBlade>().returnFromThrowPushback = true;

GameObject.Find("LooseCopterBlade1").GetComponent<looseBlade>().offsetStartingTime = Time.time;

GameObject.Find("LooseCopterBlade1").GetComponent<Renderer>().enabled = true;

if(goLeft)

{

GameObject.Find("LooseCopterBlade1").GetComponent<looseBlade>().leftReturn = true;

}

else

{

GameObject.Find("LooseCopterBlade1").GetComponent<looseBlade>().leftReturn = false;

}

}

if (identityValue == 2)

{

GameObject.Find("LooseCopterBlade2").GetComponent<looseBlade>().returnFromThrowPushback = true;

GameObject.Find("LooseCopterBlade2").GetComponent<looseBlade>().offsetStartingTime = Time.time;

GameObject.Find("LooseCopterBlade2").GetComponent<Renderer>().enabled = true;

if (goLeft)

{

GameObject.Find("LooseCopterBlade2").GetComponent<looseBlade>().leftReturn = true;

}

else

{

GameObject.Find("LooseCopterBlade2").GetComponent<looseBlade>().leftReturn = false;

}

}

if (identityValue == 3)

{

GameObject.Find("LooseCopterBlade3").GetComponent<looseBlade>().returnFromThrowPushback = true;

GameObject.Find("LooseCopterBlade3").GetComponent<looseBlade>().offsetStartingTime = Time.time;

GameObject.Find("LooseCopterBlade3").GetComponent<Renderer>().enabled = true;

if (goLeft)

{

GameObject.Find("LooseCopterBlade3").GetComponent<looseBlade>().leftReturn = true;

}

else

{

GameObject.Find("LooseCopterBlade3").GetComponent<looseBlade>().leftReturn = false;

}

}

if (identityValue == 4)

{

GameObject.Find("LooseCopterBlade4").GetComponent<looseBlade>().returnFromThrowPushback = true;

GameObject.Find("LooseCopterBlade4").GetComponent<looseBlade>().offsetStartingTime = Time.time;

GameObject.Find("LooseCopterBlade4").GetComponent<Renderer>().enabled = true;

if (goLeft)

{

GameObject.Find("LooseCopterBlade4").GetComponent<looseBlade>().leftReturn = true;

}

else

{

GameObject.Find("LooseCopterBlade4").GetComponent<looseBlade>().leftReturn = false;

}

}

Destroy(gameObject);

}

\*/

-use rend.material.color = new Color(1.0f, 1.0f, 1.0f, 1.0f); to change an object’s RGB and A (transparency)

-Use sorting layers and order in layer on an object’s sprite renderer to customize what objects appear in front of others

-MUST have using UnityEngine.SceneManagement; at the top of a script to be able to load a scene!

-When using Physics2D.IgnoreCollision, and the object you are trying to ignore has multiple child objects with box colliders, these box colliders must ALSO be separately made to ignore the object in question, otherwise it won’t work!

May 10:

-To find a bunch of GameObjects in Unity that all share the same tag, you need to first create a GameObject array:

Public GameObject[] array;

Then, you find all the objects:

Array = GameObject.FindGameObjectsWithTag("Blade");

And then, you use a Foreach loop to cycle through each part of the array, and do something with it:

foreach(GameObject blade in bladeArray)

{

Physics2D.IgnoreCollision(blade.GetComponent<BoxCollider2D>(), this.GetComponent<BoxCollider2D>());

}

-Note that the part after GameObject (“blade”) is made up right there, you don’t have to declare it earlier

-To make two objects collider with each other again after using IgnoreCollision, put “false” at the end of the statement:

Physics2D.IgnoreCollision(groundObject.GetComponent<BoxCollider2D>(), this.GetComponent<BoxCollider2D>(), false);

May 12:

-Whenever you create a new code, it will likely have inconsistent line endings. You need to make them consistent, otherwise Unity will severely lag

-IN ORDER TO CORRECT THIS, in Microsoft Visual Studio, you must go to File>Advanced Save Options>Line Endings>Windows

-NEED TO ADD A RIGIDBODY to an enemy if I want Unity to continuously register that I am touching it. If I don’t have a Rigidbody attached to it, then “OnCollisionEnter2D” or “OnTriggerEnter2D” won’t work

May 16:

-When using a spring joint, set the damping ratio to 1 so that the spring does not go up and down a lot when pressure is applied (oscillate)

-In order to make the spring jointed object get pushed further down when jumped on, decrease its mass

May 18:

-When trying to make the player “stick” to a platform, make the player a child of the platform, so that their position becomes tied to the platform’s position:

//Makes the [GameObject](https://docs.unity3d.com/ScriptReference/GameObject.html) "newParent" the parent of the [GameObject](https://docs.unity3d.com/ScriptReference/GameObject.html) "player".

player.transform.parent = newParent.transform;

-Update: good for simple timers, non-physics based objects, and receiving input

-FixedUpdate: good for extremely precise timers, physics based objects (rigidbody)

-Update is not called on a regular timeline. If a frame takes longer to process than the next, then the time between update calls will be different. FixedUpdate is called on a regular timeline, and is called the same time between calls

-FixedUpdate is **NOT GOOD for getting Key Input!**

May 23:

-If you make the player a child of a gameobject you want to go on top of, this will eliminate the weight the player is applying to the gameObject in question

-If you don’t make the player a child, then the player will weigh down the platform object, which will create problems if the platform is programmed to go up and down

July 8:

-When I have to make an object a rigidbody in order for its collision to be detected, but want to fix its position in place, freezing it’s constraints isn’t enough – I also need to increase the object’s weight, so that other objects won’t weigh it down.

July 23:

-When you want an object’s collision to be detected, at least one of the objects needs to have a rigidbody

-To destroy the parent of a gameObject, you must do: Destroy(transform.parent.gameObject)