Unity notes

* Press F to focus perspective on the selected object
* Add Rigidbody for in game physics
* Ctrl+B for build settings
* Player input settings
  + Create Input Action Asset and link certain keys to an action name
  + Add the new script component to an object
* Writing scripts
  + A function is a way to group code under one name
  + Two functions included in the template, Start and Update
  + Update is called before rendering a frame and this is where most of your code will go
  + Fixed update is called just before performing any physics calculations and this is where your physics code will go
  + Name spaces are a collection of classes and other datatypes which can be imported at the start of your script
  + Already three name spaces in the template: System.Collections, System.Collections.Generic, UnityEngine
  + Add a new one called UnityEngine.InputSystem; for inputs
  + Void means this function will do a task and won’t return any values
  + Write void OnMove()
  + Inside the parentheses, add InputValue (type of variable) and movementValue (name of the variable you will use in the function)
  + The movement value variable will capture and store the data from the x and y direction of input, this kind of data can be stored as a vector2 variable
  + Write Vector2 movementVector = movementValue.Get<Vector2>() in the parentheses; this code takes or gets the vector2 data from the movementValue and stores in the vector2 variable you are creating called movementvector
  + Write private Rigidbody rb; This will create a private variable of the type rigidbody and call that variable “rb”,
  + The variable is private and not public because you don’t need the variable to be accessible from the inspector or other scripts right now
  + Next, inside the Start Function, write rb = GetComponent<RigidBody>(); This sets the value of the variable rb by getting a reference to the rigidbody component
  + Add new function called void FixedUpdate() below the start function
* Code for adding force
  + In the FixedUpdate function, add rb.AddForce(movementVector);
  + There are three vectors x, y and z
  + You need to create two new variables for individual input actions
  + Underneath the Rigidbody variable, add private float movementX; and
  + private float movementY;
  + In OnMove, add movementX = movementVector.x;
  + MovementY = movementVector.y;
  + Write in the FixedUpdate, Vector3 movement = new Vector3(movementX, 0.0f, movementY); The f signifies that this is a float value
  + Remove Vector from the rb.Addforce(movementVector);
* Code for movement speed
  + First, add a speed variable at the start, public float speed = 0;
  + Back in fixedupdate, add \* speed to the addForce function
* Moving the camera
  + Move the camera under the player in hierarchy so that the camera will move alongside the player
  + But the camera will also roll as the player roll
  + Write a script for the camera because the above method doesn’t work
  + Add public GameObject player; which will reference the player object
  + Add another variable declaration private Vector3 offset; this will store the offset value,
  + Take the current transformed position of the camera game object and subtract the transformed position of the player game object to find the difference
  + Add offset = transform.position - player.transform.position;
  + You will use that to camera game object’s transform position and this needs to happen every frame so write it in update function
  + transform.position = player.transform.position + offset;
  + Now, when the player moves, the frame before displaying the camera, the camera game object is moved into a new position aligned with the player game object before the frame displayed.
  + This works because only the position is added and not the rotation
  + However, you don’t control the order all of the update functions happen, that means the update can run before other scripts
  + The solution is to add LateUpdate which will run after all other scripts
  + You have to create a reference to the player game object \*\*\*\*
  + Drag the player component to the camera player tab
* Setting up the play field
  + Create an empty game object and name it walls
  + Create a cube named west wall and place it under walls
* Creating pick up objects
  + Create a cube, make it small, rotate it and color it
* Rotating the pickup object
  + Remove the start function from the template
  + The rotate values need to change every frame
  + There are two ways to transform a game object: translate (moves the game object by transform) and rotate (rotates the game object)
  + It has two possible parameters: Using a vector3 variable or 3 float variables xyz
  + Add transform.Rotate(new Vector3 (15, 30, 45) \* Time.deltaTime); make sure the “t” is lowercase because you are referring to the component not the variable type
  + Deltatime is perfect for smooth action because it’s a float that represents the differences in seconds between the last frame, multiplied by per second rather then per frame
* Making the pickup object into a prefab
  + A prefab is an asset that contains a template or a blueprint of a game object or a game object family
  + All the prefab objects will be updated with just one
  + Drag the pickup object from the hierarchy into the prefab folder in project
  + When you drag something from the hierarchy into the project window, unity creates a prefab asset containing a template or blueprint of the game object
  + The game object will turn blue when it becomes a prefab
  + Create an empty object and put the pickup object underneath it
* Detecting Collisions with collectibles
  + You will use the Ontrigger function on the player script
  + Write private void OnTriggerEnter(Collider other)
  + Other is the identifier that the sphere will hit
  + In the body, write other.gameObject.SetActive(false); this code will disable game objects correctly
  + Tags allow you to identify game objects by comparing tag values to a string
  + First, set a tag value for the pickup object
  + Add a tag called PickUp and apply it to the pick up object
  + Above the ontriggerenter, write if(other.gameObject.CompareTag(“PickUp”)) {}
  + Comparison is case sensitive \*\*\*\*
  + Paste the other code into the curly brackets
  + Then make the collider into trigger colliders Ex. A player can run into a door and a message will open that says you have discovered a new route or every time a player walks around a particular corner, spiders will drop from the ceiling
  + Any game object with a collider and a rigidbody is considered dynamic which means unity won’t treat it as static
  + If the pickup object is static, it will take unity longer each frame to calculate the physics and it would do it each frame because they are rotating
  + Add a rigidbody component to the pickup objects
  + The use gravity checkbox can prevent them from going down
  + Enable the ls kinematic (A kinematic rigidbody will not react to physic forces)
  + Standard rigidbodies are moved by physics forces and kinematic rigidbodies are moved by transform
* Adding score
  + Add a variable for integer count – private int count;
  + Adding an initial amount for count variable – write under Start function count = 0;
  + On the OnTrigger function, count = count + 1;
  + User Interface and Text Mesh Pro
  + Add text mesh pro from hierarchy, the important thing is that all UI elements must be the child of the Canvas behaved correctly
  + Rectangular transform for the text, Hold shift and alt to get it in the left corner
  + Before coding, you need to add information about text mesh pro in the script
  + The details about the UI elements are held in the name space
  + Write a new line called, using TMPro;
  + Under the public variable called speed, create a new variable called
  + public TextMeshProUGUI countText;
  + Underneath the OnMove function, write void SetCountText() {}
  + Write countText.text = “Count:” + count.ToString();
  + Under start function, write SetCountText();
  + Inside the ontrigger function, write SetCountText();
  + Drag the counttext object into the player component
  + Go to the event system object and replace the shit
* Creating a Game end message
  + Create a new textmeshpro object and transform the text whatever
  + Now, you need to add a reference for the UI text in the script
  + With the other variables, write public GameObject winTextObject;
  + Set the starting state to be disabled
  + Under Start function, write winTextObject.SetActive(false);
  + Underneath the setcounttext function, write if(count >=12) {}
  + In the curly brackets, write winTextObject.SetActive(true);
  + Don’t forget to drag the gameobject into the player component

Citations:

<https://answers.unity.com/questions/1255990/how-to-change-the-color-of-an-object-when-it-colli.html>

https://answers.unity.com/questions/1427037/how-to-generate-a-random-color.html

**Needles to slay tutorial**

Variables

* When creating a variable, define the type of variable and name the variable.
* Types of variable
* Whole number = integer
* Line of text = string
* A semicolon defines an end of an instruction
* That one variable can have different assigned values within the same script, because the computer executes instructions sequentially
* <https://docs.unity3d.com/ScriptReference/index.html> Scripting reference
* Save state so that you can restore any changes after
* // is for comments

Functions

* Void AddingNumbers(float num1, float num2)
* Void means no type or empty because its not stored anywhere
* Adding Numbers is the name of the function
* Parentheses are required even if the function takes no variables
* Floats are fractional numbers, add an “f” after a number to make it float
* Function call has a function name, set of parentheses containing two values and semicolon
* Change the parameters in the function to substitute the variables
* The curly braces {} are define scope. That means the only variables declared inside the function exist there.

Classes

* A class enables you to group together related functions and variables.
* Subclasses have access to the features of the parent class
* Use dot operators (a full stop) to access different variables within a class.
* Class as a type
  + Vector3 myVector = new Vector3();
  + New means to create a new variable space in memory
  + It’s only when you want to create a new Vector3 variable that you need to use that new syntax!