NOTE: gameObject refers to the game object holding the script, GameObject refers to the game object data type

scripts can only change fields of things if they have a reference to the thing

* 
* 

running a function manually while game is running

* Text

  Description automatically generated
  + context menu adds option to run the function
  + Graphical user interface, text, application, chat or text message

    Description automatically generated

UI functions can only be accessed after importing the UI part of the UnityEngine in preprocessor

* 
* 

Triggers

* invisible objects with colliders
* used to run events when an object enters it
  + Graphical user interface, application

    Description automatically generated
  + Graphical user interface

    Description automatically generated with medium confidence

Difference between Start and Awake

* Awake()
  + gets called immediately when an object is instantiated
  + interrupts code flow to run
* Start()
  + called at the start of the frame AFTER an object is instantiated

Saving data with PlayerPrefs

* 
* Text

  Description automatically generated

Destroy all instances of an object

* create array of all objects of a specified type with FindGameObjectsWithTag
* iterate through array and destroy all objects
* Graphical user interface, text

  Description automatically generated

Time based vs frame based

* aka untying stuff from framerate
* Text

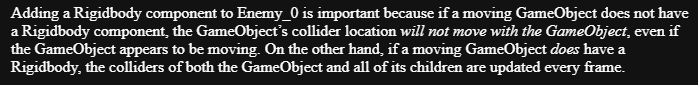
  Description automatically generated
* multiply a movement value by time.deltatime
* time.deltatime is time since last frame
  + adding up x \* time.deltatime means that after 1 second it would move x units
  + allows for use of bigger, easier to understand numbers
  + makes stuff in update() method (dependent on framerate of machine) behave like in fixedupdate (60x per second)

Scale of objects in Unity

* meters
* most modeling programs use centimeters, so multiply scale by .01 to equate it
* Text

  Description automatically generated

Colliders will NOT move with an object if it does not possess a rigidbody

* 
* rigidbody collision can be disabled, it just has to have one
* also applies to children objects (like groundDetector)

Enums

* states accessed by enumName.state
* set by enumName = enumName.state;
* 
* A screenshot of a computer

  Description automatically generated with medium confidence
* Text

  Description automatically generated

GetComponentInParent<>()

* first checks the CURRENT game object, then checks the one above it, then the one above that, etc. until a matching component is found
  + means that if the child and parent both have a matching component, itll return the childs component like getComponent<>() does
  + can be fixed by calling transform.parent.GetComponent<>()
    - gets the parent of the transform, and gets the component from that object

image compression on Texture 2D

* only compress big images
  + small images look REALLY bad when compressed
* filter mode
  + automatic blurring from unity
  + set to **point** to turn it off and see compression issues clearer
  + set back to **bilinear** when compression issues are solved
* compression
  + compresses image to make it smaller
* alpha source
  + set to none to shrink images
    - tons of size is unity storing unneeded alpha information
    - only do this if image has no transparency
* crunch compression
  + gives really small image with some quality loss
  + best used with normal compression

ui with canvas and textmeshpro

* canvas scale is in pixels
* normal unity scale is in meters
* setting ui to scale to any res
  + canvas>canvas Scaler
    - UI scale mode: scale with screen size
    - reference resolution: res of game pane
    - match: all the way to right to match height

Tilemaps

* <https://learn.unity.com/tutorial/introduction-to-tilemaps>
* tilemaps only collide with rigidbody 2d’s

Layers / Sorting layers

* Layers
  + for physics interactions
  + set in project settings (matrix thing)
* Sorting layers
  + for different layers of sprites rendering on top of eachother
  + changed in sprite renderer > additional settings
  + set in project settings

Animations

* drag sprites that comprise an animation into the hierarchy to make the game object for the player and the animator, then add new clips to the newly created animator
* example
  + Text

    Description automatically generated

Top down movement

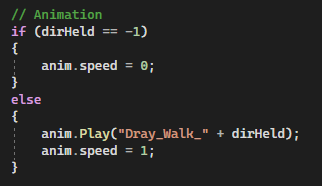
* example
  + Text

    Description automatically generated
* 2nd example
  + Text

    Description automatically generated
    - inputs change the dirHeld variable, used to access vector2’s corresponding to directions in the directions array
* 3rd example
  + Text

    Description automatically generated
  + allows for modular input keybinds

playing animations programmatically

* 
  + plays the animation from the animator’s clips of walk\_1, walk\_2, etc

Art

* <https://www.reddit.com/r/gamedev/comments/131oo3c/a_programmers_guide_to_learning_game_art/?utm_source=share&utm_medium=android_app&utm_name=androidcss&utm_term=1&utm_content=share_button>

Events

* Objects can react to an event raised by a different object
* Cannot pass data, only a mechanical reaction to an event that occurs
* Negatively impacts performance, good for rare events but for often occurrences it will tank frames
* Good for having many different objects react to a single occurrence
  + Like all objects resetting upon player death
* Either use unity system or build a custom framework
  + Costly and obnoxious imo
* VERY similar to just using a reference to another script to call a method
  + Instead of script 1 having a reference to script 2 that it uses to call a method…
  + Script 1 has an event that holds a reference to a listener on script 2, that then calls a method on script 2

Singleton

* Script1 holds a static field of Script1, that it will check when a new Script1 is created
  + If the static field is null, there are no existing Script1’s. it will set the static field = this
  + If the static field is not null, there is already a Script1. The script will then delete itself

Events + Singletons

* Example
  + Gamecontroller singleton has coinCollect event
  + Coin object invokes the coinCollect event on the singleton
  + The event then triggers everything that should happen when a coin is collected
    - All other relevant objects are listening to the SINGLE event on the singleton, instead of a BUNCH of different events on all the different coins
    - Also makes registering the event easier because singleton
* Good for:
  + Having a bunch of stuff happen when a single thing happens (effects, score updates, etc)