Scripting II:

Tues 1/11—Inheritance

Thurs 1/13—Inheritance cont. (covers AI); overriding, abstract classes, virtual functions

Tues 1/18—Inheritance cont.; abstract classes vs concrete classes; interfaces; casting; fields and properties

* Recommends c# pocket reference book (can find pdf online)
* Interfaces force the same variables, set up, functions; it forces us to code a certain way
  + Good for teams and for making things similar and readable
  + Makes scripts compatible
  + Are accessible by any class—don’t need children
  + \*interfaces should always start with “I” (IHealth, IMove, IPowerUp, etc)

Thurs 1/20—animation/character controller issues, scripting with physics, get new game ideas:

* Open phone and play more games
* Watch more shows
* Draw every day
* Have self-esteem…aka don’t be afraid of failure and take some risks
  + Abcdefu is your new anthem

also, when you wanna pause your game, script your rigid bodies to go to sleep

tues 1/25—pooling; created gun/bullet pool in unity; toward end we used game actions to make a self-populating pool; changing script execution order in settings

* Anthony’s code is on 2670 fall 2021, named “poolingbehavior” and “bulletbehavior” (or it should be if it pushes to git; if not, the code is visible in the video)

Thurs --??

Tues 2/1—waves of enemies, static variables, replacing scenemanager with better code, playerprefs and datastorage script (in anthony’s repo)

* Scene manager destroys the level every time you unload it, creating memory leaks and crashing the app over time

Thurs 2/3—game actions, events, and handlers

Tues 2/8—a way to replace scenemanager using a vector 3 data to reset the transforms of your objects; how to make a branch in gitkraken, more unity actions, a little bit of particle system stuff

* You can actually use this vector 3 data as checkpoints
* He did mention that you sometimes have to load in scenes if large changes need to be made, but if you only need to reset your positions, this is a better way to go

Thurs 2/10—game actions

* Watch “the scriptable object revolution” on youtube by unity
* Here are the basic steps needed to run a game action
  + You have 2 objects, for example a button and a score obj. When the button is pressed, you want to update the score obj. The button is the one to shout out (“I’m pressed”) and the score is the one to listen.
  + You have a game action scriptable object and a game action handler script (see lab or photo app or anthony’s repo for these scripts). Make sure the game action SO has [CreateAssetMenu] on it.
  + First, make a new game action SO (create>game action) for this action.
  + In the button script (the shouter!) you need a unity event. Buttons already have one built in (it’s called onclick in the editor). Pull up the event in the editor, click +, and drag the new game action SO you made into the box. Select “Raise()” from the dropdown.
  + In your score obj script, you need to have your function written. For this example, we would have an “UpdateScore()” function. But you don’t need any events or anything special IN the script itself.
  + Click on the score obj in the hierarchy. Drag and drop the game action handler script onto it. (It needs the handler since it is the one listening!) In the editor, in the new Game Action Handler script, do this:
    - For the Game Action Obj slot, drag your new SO
    - In the respond event, click +. Here is where we update the score! Drag the score obj to the box slot and select “Update Score()”
  + Now play test. When you press the button, the score obj should now update!
  + Sometimes it’s helpful to add a debug.log to the game action function “Raise()” so you can see if it is triggering or not when you click. If it’s not working, double check your game action SO and handler script. And debug.log everything!
  + TLDR; there’s a caller/shouter obj and a receiver obj. The shouter has the game action SO, the receiver has the game action handler. The shouter has an event, the receiver doesn’t. The end.

Tues 2/15—in app purchases/store, casting, interfaces, parameters over fields (different kinds of basic variables), creating art class inheritance

* Parameters control what gets/sets the var, but it isn’t visible in the unity editor…in order to make it visible, right click on the var in rider and create backing field (? I think that’s what it’s called)

Tues 2/22—overloading (setting method overloads with multiple functions)

* Debug.Log() is a great example of this; it has several functions of the same name with different parameters

Thurs 2/24—character controller, generics

Tues 3/1—input system…a new system that works without using update()

Thurs 3/3—new input system (watch this lecture, it’s much more coherent)

Tues 3/29—new input system (best one so far)

Thurs 3/31—game states, enums, audio control

Thurs 4/14—self populating SOs