Summary of the SCRIPTS folder

1 unity unit of space = 1 meter

Search Tags:

#TODO-CODE: change code

#TODO-DOC: write documentation

* **Character**: scripts associated with the player character
  + **Animation**: Controlling character animations
    - **class VelocityToAnimatorController:** Links the Character Controller and animator. Designed for use with Dhillon’s animator controller.
  + **class CharacterControllerForce**: Allows the CharacterController to bump into rigidbodies. Kind of buggy right now and all values are hard coded
  + **class Energy**: controls Dhillon’s cumulative energy
    - **Max Energy**: the numerical amount of energy Dhillon can hold at max
    - **Starting Energy Ratio**: Dhillon starts the level with this fraction of his max energy.
    - **public void IncreaseEnergy(float)**: add this much to Dhillon’s energy
    - **public void DecreaseEnergy(float)**: remove this much from Dhillon’s energy
  + **class MouseRotation**: Turns the character and camera with the mouse.
    - **Max Angle Vertical**: Max angle, in degrees, that the character can tilt the camera up or down
    - **Turn Speed**: Mouse sensitivity. Higher values mean more sensitive.
    - **Target Character**: The root of character this is attached to. Ideally, this will be set in the character prefab.
  + **class PlayerCharacter**: Controls the player character’s motion.
    - **Run Speed**: Run speed in meters/second
    - **Gravity Scale**: Acceleration due to gravity is 9.8 meters/sec^2 times this value. Despite 1 being physically realistic, it feels kind of floaty.
    - **Jump Strength**: Initial jump velocity is this value /sec upwards
  + **class PlayerSprint**: goes on the root character. Lets you sprint #TODO-CODE: this does not reduce energy or wear Dhillon out
    - **Sprint Key**: the key you hold to sprint forward
* **Debug**: Generally hacky stuff that won’t be used in the final version
  + **static class OnScreenConsole** Allows you to print messages directly to the screen. This is tied to a GameObject that will be automatically created if you use any of the methods or properties in this static class.
    - **private Color m\_textColor**: the color the text appears in on screen. Serialized so you can change it on the GameObject
    - **static Color textColor{get, set}:** encapsulates m\_textColor of the static singleton.
    - **static void Log(object)**: print the string representation of an object to the lower left hand side of the screen. Old messages move upward.
    - **static void ClearConsole()**: clear the on screen console
  + **class RandomStuff**: Stray methods that need to be instance methods to be tied to UnityEvents.
* **Editor**: scripts that control the behavior of the Unity editor. (Editor is a keyword folder name)
  + **class OpenCSharp:** patches a bug where VSCode does not open the project as the workspace.
    - **MenuItem “Open C# Project in Code”:** Windows only. If you don’t use VSCode, aren’t running Windows, or don’t have the VSCode bug, don’t click this.
  + **class ScriptableObjectCreator:** creates a blank instance of any ScriptableObject
    - **MenuItem “Create/Instance”:** Right click on the .cs file of a ScriptableObject and it will create a blank instance of it in the same folder
* **Objects**: Physical objects in the game
  + **General**: abstractions and all-purpose objects
    - **abstract class Gazeable:** Something that glows when moused over. If you want to make an interactable that requires something other than this, extend this class.
      * **bool currentlyInGaze{get;}:** is this object currently moused over?
      * **protected virtual void OnGazeEnter ()**: called when the object is moused over
      * **protected virtual void OnGazeExit ()**: called when the object is moused over
    - **abstract class Interactable:** Something that can be clicked on to interact with.
      * **abstract void Interact ():** what happens when you interact
      * **abstract string promptText { get; }:** text that appears over the object when gazed at
    - **class InteractableEventObject:** Interactable that can be customized in the Unity editor
      * **Text:** encapsulated by promptText
      * **On Interact:** what happens when you interact
  + **Testing:** objects that are not relevant to the game and are only for testing
    - **class BallGenerator:** Spawns balls when you interact with it.
    - **class BallObject:**  Interact to kick the ball. Works even at a distance.
    - **class GazeCube:** Prints its name to the on screen console when you gaze at it.
    - **class NewspaperReadableObject:** Interact to enable the Newpaper UI element
      * **Newspaper UI:** GameObject that has the newspaper ui
    - **class ThinkAboutSomething:** make Dhillon think about something
      * **Subject:** The world header is “Think about \_\_\_\_”
      * **Message:** This is what Dhillon thinks
* **Timer**: Scripts related to time
  + **class Daylight**: Controls the timer ticking out. #TODO-CODE: rename
    - **Depletion Scale**: Timer ticks at this rate. 1 is realtime, 2 is twice as fast.
    - **Initial Duration**: The level starts with this many seconds on the clock.
    - **Daylight UI Bar**: the DaylightUIBar component controlled by this object. Should only be one per scene.
  + **class DaylightUIBar**: UI representation of the timer
    - **Clock Rect**: RectTransform of the moving clock
    - **Image**: bar background for the timer #TODO-CODE: rename
    - **Start Pos, End Pos**: the positions at the beginning and end of the timer, respectively. It’s a pain to set these manually. Use the ContextMenu items instead.
    - **ContextMenu “SetStartPosition”, “SetEndPosition”**: Set the current anchored position of the clock to the desired start/end position
    - **ContextMenu “SnapToStartPosition”, “SnapToEndPosition”**: Set the current anchored position of the clock to the start or end position
* **UI**: UI related scripts
  + **ThoughtBubble**: Dhillon’s thoughts on things in the world
    - **class RolloutText**: put this on a UI text to have the characters roll out one at a time on enable
      * **Speed:** the text rolls out at this rate (characters per second)
      * **Message:** The message that will be displayed when enabled
      * **Next Button:** An icon to appear when the text is done rolling out.
    - **class UIEllipticalMotion:** #TODO-DOC
    - **class ThoughtBubble: #**TODO-DOC
  + **class EnergyBar**: visualization for Dhillon’s energy bar #TODO-DOC
  + **class ObjectCenteredBillboard:** #TODO-DOC
  + **class OffscreenUIElement:** For UI elements that are intended to slide on screen when enabled.
    - **Off Screen Pos**: anchored position while off screen
    - **On Screen Pos**: anchored position while on screen
    - **Transition Time:** time it takes to transition for off screen pos to on screen pos
    - **Screen Darken Factor:** While active, the screen darkens behind the UI element by this factor. 0 is completely unobscured, 1 is fully black.
    - **Interpolation Method**: Determines the entrance animation.
  + **class TextHeaderForWorldObject:** Makes a UI text follow the screen position of a 3D world object. This is used for the text prompt on Interactables.
    - **void SetText(string):** changes the text
* **Utility**: Classes and methods for convenience.
  + **class ColorHSV**: HSV representation of a color. Converts implicitly to Color
    - <https://gist.github.com/cjddmut/fefe5dac35cccfceabec>
  + **class Interpolation:** More flavors of Lerp #TODO-DOC
  + **enum InterpolationMethod:** Enum for various types of interpolation functions for use with Interpolation.Interpolate(..) #TODO-DOC
  + **class Utility**: #TODO-DOC #TODO-CODE: move this into separate extension classes, import from old projects
* **class Game**: #TODO-DOC
* **interface Initializable**: All initializables will have their Initialize() called during Game.staticRef’s Start() #TODO-DOC: improve