Jason Frehner

Notes

While the test's primary goal is to see how a developer translates prototype design into a functional prototype within Unity, this exercise illustrates a few other underlining goals.

- How is prototype code designed in a way to last beyond the prototype?
  - Here we have isolated the systems and other minor functions with only thin connections between them where needed.
- How are non-designed or undocumented aspects implemented?
  - The enlarging of an active button was not called out, but the design showed that when it was active the button was larger.
  - Tasks like this are implemented in a manner that makes them easy to iterate on or completely remove if needed after review.
- How is a new system communicated?
  - Code is kept clean with the intention of being easily readable, and other elements of this document serve as a general outline of the systems intent. I hope between this document and the submitted work I have addressed all of these issues.

Assumptions

- This is not final Animation or UI.
  - How easy will it be to change things?
    - A prefab was created for the buttons.
    - Button images are split and white based, so some updates like colors do not require new assets.
- We want to prototype functionality.
  - The key functionality being the UI sliding for the jump kick.
- We don't want code wasted, so as much as possible make code that can live beyond the prototype.
  - Keep UI logic separate and make components for any reusable functions.
- We want, more flexibility over the jump, so the system should control the Y transition.

System Design

The main point here is the decoupling of the UI from the rest of the system.
Some benefits are:

- Each system can be built out independently.

- Logic is separated, nothing about the animation controller is known at the UI level.
- Easy to expand to other systems, any new system that needs user input would not be mixed in with animation controller logic.

The Player Manager and additional jump animation logic are also seperated from each other, keeping their focus and individual focal small.

UI

- One element not called out and easy to miss in the Figma doc, is that the jump button while held down is larger.
    - RectAction.cs give us this ability to adjust an active button's size.
- The obvious way of grabbing all the button icons is with the icon flattened into one image form Figma, however this limits the flexibility of UI in Unity.
    - I have seperated the background and foreground icon images. This hase several benefits like:
        - Dynamically controlling the colors of each element independently.
        - Ability to animate the inner icon.
        - Reduced assets, no need to use different disabled or enabled images.
        - More possibilities with UI shaders.

Animation

A common approach is to have the player manager also act as a state machine that mimics the same states we have in the Animator state machine. So that additional control (jump height) can be through this state machine.

However, using an Animator State behavior script saves us from having to maintain two state machines. And help us to isolate our code into easier to manage sections.

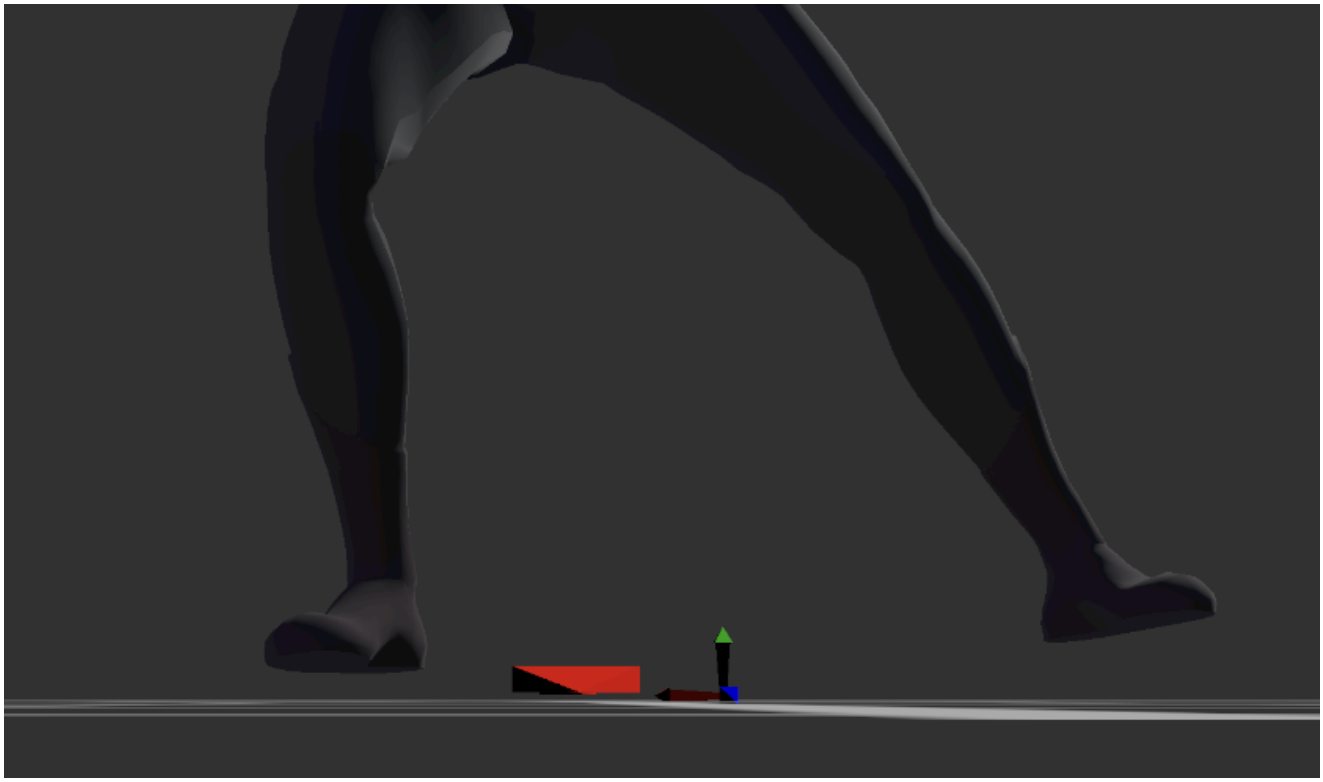
Questions

Some questions or concerns that I would bring up to other team members. In the case like this where I don't have immediate feedback and nothing is a major road block for getting an initial iteration for review I have made my best judgement and made things easy for future changes.

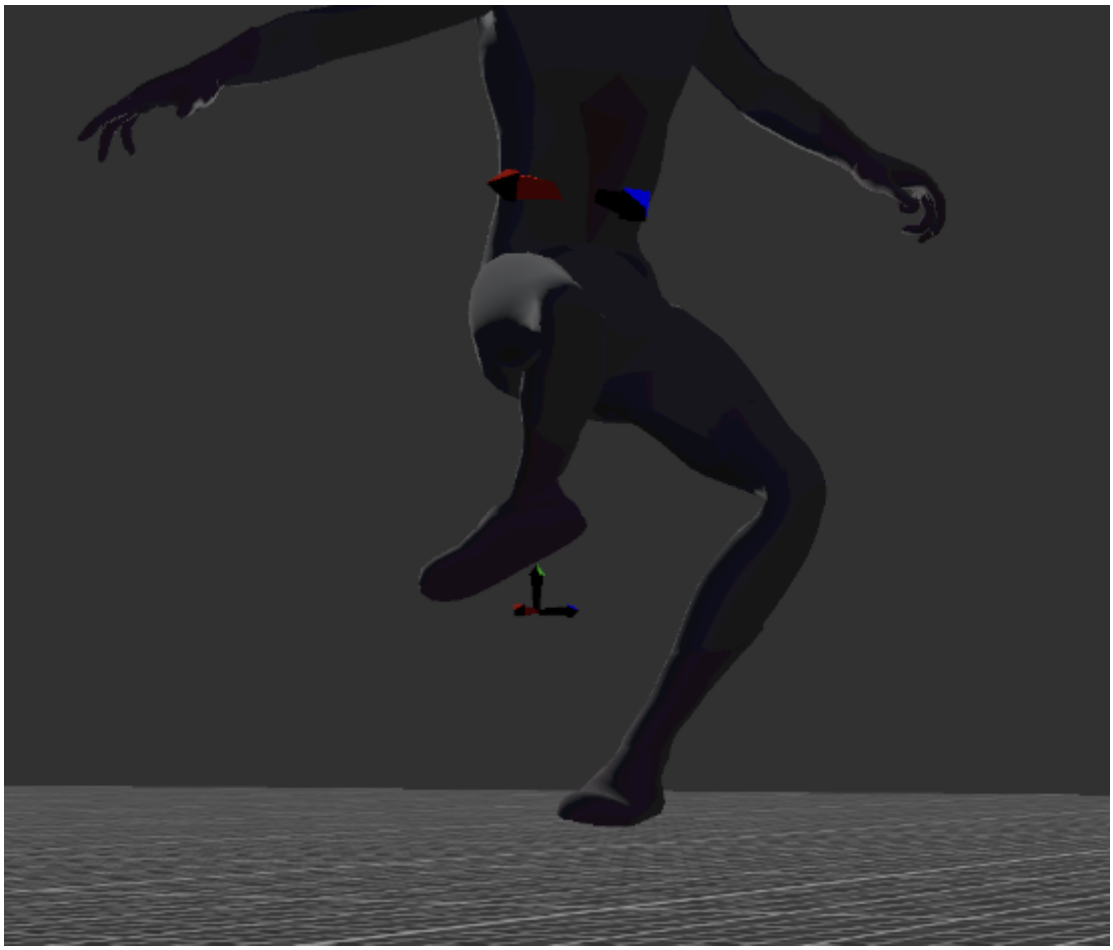
For Animator

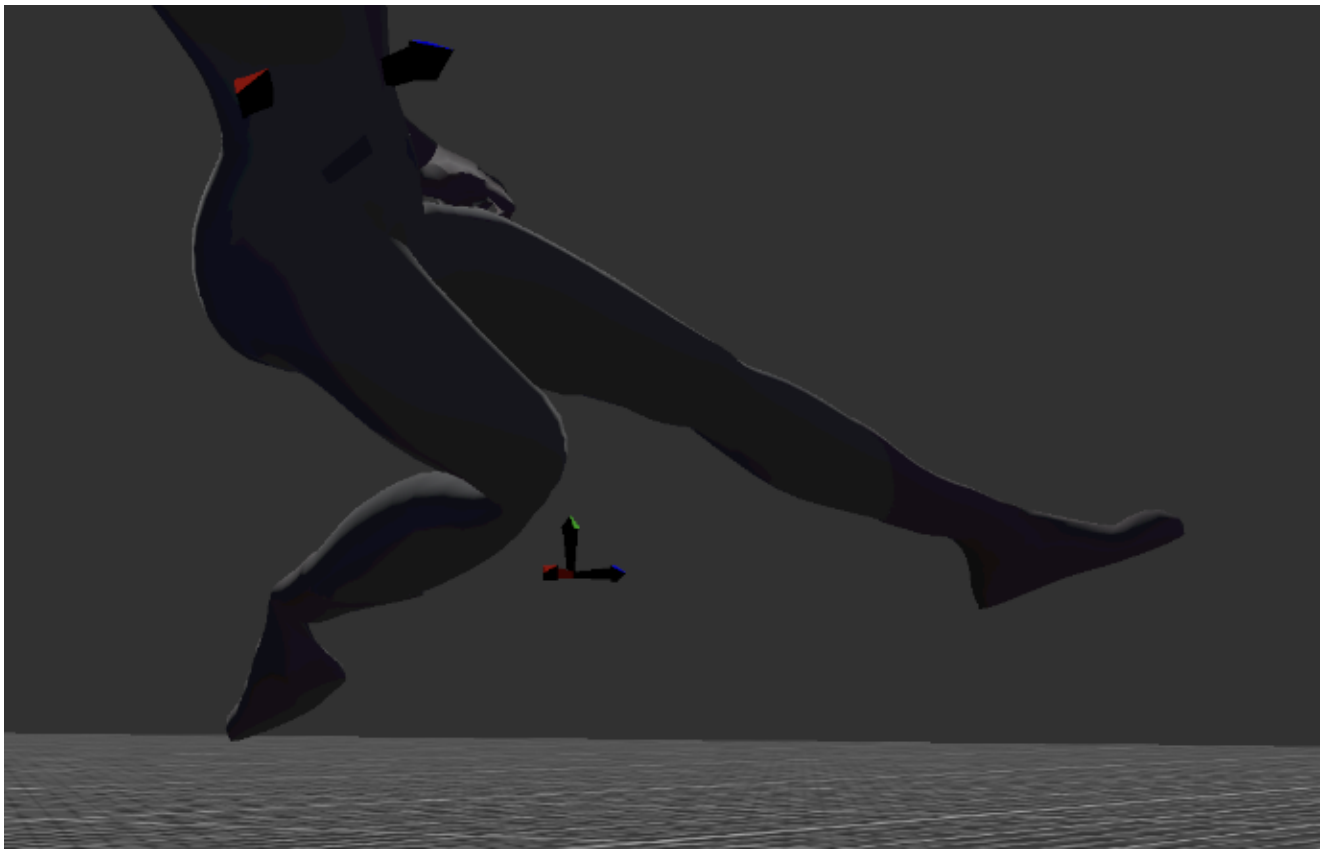
- The connections between animations, while thought to be obvious don't always seem to blend well.

- Idle animation's feet are not firmly on the ground.



- Jump Up Ascend, Descend, Idle and Kick have broken legs / feet.





- Let's connect about planning out transition animations.

For UI/UX

- Is UI positioning Relative or Absolute?
- Is the jump button only larger because it is active?
    - Will other buttons be this large?
- To speed things up, having a section in Figma that is the icons split apart with separate background and foreground, and all white based. In addition a color swatch to easily grab all the hex codes would be very helpful and a useful reference.
- What is the intended UI flow of the slide from jump to kick button, does the slide area draw as the user slides their finger?
- Should the character jump out of the screen or should the camera follow?

Unity Packages to Consider

Motion Matching
DOTween Pro